MUX Granularity Oriented Iterative Technology Mapping for Implementing Compute-Intensive Applications on Via-Switch FPGA

Takashi Imagawa[†1] Jaehoon Yu[†2] Masanori Hashimoto[†3] Hiroyuki Ochi[†1]
†1 Ritsumeikan University, Shiga, Japan, †2 Tokyo Institute of Technology, Tokyo, Japan
†3 Osaka University, Osaka, Japan
{takac-i,h-ochi}@fc.ritsumei.ac.jp, yu.jaehoon@artic.iir.titech.ac.jp, hasimoto@ist.osaka-u.ac.jp

***Abstract*—This paper proposes a technology mapping algorithm for implementing application circuits on via-switch FPGA (VS-FPGA). The via-switch is a novel non-volatile and rewritable memory element. Its small footprint and low parasitic RC are expected to improve the area- and energy-efficiency of an FPGA system. Some unique features of the VS-FPGA require a dedicated technology mapping strategy for implementing application circuits with maximum energy-efficiency. One of the features is the small ratio of logic blocks to arithmetic blocks (ABs). Given an application circuit, the proposed algorithm first detects word-wise circuit elements, such as MUXs. These elements are evaluated with an index of how resource utilization and fan-out change when the corresponding element is implemented with AB. All these elements are sorted in descending order based on this index. According to this order, each element is mapped to AB one by one, and synthesis and evaluation are repeated iteratively until satisfying given design constraints. The experimental results show that resource utilization and maximum fan-out can be reduced by about 30% to 50% and 12% to 87%, respectively. The proposed algorithm is not limited to the VS-FPGA and is expected to improve computation density and energy-efficiency of various FPGAs dedicated to compute-intensive signal processing applications.**

***Index Terms*—SRAM-based FPGA, CAD, EDA, design-space exploration**

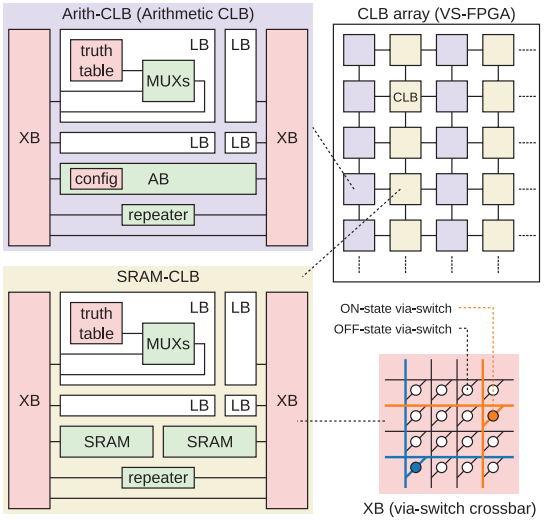

Fig. 1. Block diagram of CLB array (VS-FPGA). The red and green components are implemented on BEOL and FEOL layers, respectively.

I. Introduction

FPGA is a well-suited platform for implementing various computing systems, including state-of-the-art machine learning and signal processing algorithms. However, the achievable performance and energy efficiency are limited [1]. The low performance of FPGA originates from its field programmability. Reference [2] reports that 78% of the chip area is consumed for routing resources, and only 14% of the chip area is used for computing. The routing resources are implemented using SRAM cells, multiplexers, and/or pass transistors. They not only occupy most of the precious FEOL (front end of line) layer area that should be utilized for computing resources but also degrade performance due to their parasitic resistance and capacitance. Due to this, the chip size becomes larger, and hence the interconnect delay becomes longer. In addition, the interconnection in FPGA consists of a number of programmable switches and repeaters (buffers). Consequently, 80% of the circuit delay is occupied by interconnect delay [2].

For pursuing an energy-efficient FPGA platform, "via-switch," which is a novel non-volatile and rewritable memory element, has been developed [3]. The via-switch can be implemented using only BEOL (back end of line) layers so that FEOL layers can be fully utilized for computing. The low parasitic resistance and capacitance of via-switch are expected to improve the area- and energy-efficiency in an FPGA system. The first silicon implementation of via-switch FPGA (VS-FPGA) has been demonstrated at an international flagship conference on solid-state circuits [4]. Also, another VS-FPGA for compute-intensive applications is designed as an array of configurable logic blocks (CLBs), as shown in Fig. 1. There are two types of CLBs, that is, Arith-CLB and SRAM-CLB. Arith-CLB is composed of computing resources for random logic (logic blocks, LBs), those for arithmetic operations (arithmetic blocks, ABs), and routing resources (via-switch crossbars (XBs) and repeaters). SRAM-CLB is composed of SRAM for storing local data, in addition to LBs and routing resources.

This work was supported by JST CREST under Grant JPMJCR1432. This work was also supported through the activities of VDEC, The University of Tokyo, in collaboration with Mentor Graphics, Inc.

VS-FPGA has various features that are different from conventional SRAM-based FPGAs. In this paper, we focus on the following two points in particular. The first is the small number of repeaters provided in the routing resource. The second feature is the small ratio of LB to AB. In the aforementioned VS-FPGA designed for compute-intensive applications, the ratio of the number of LBs to ABs is 8:1. These differences require a dedicated technology mapping strategy for VS-FPGAs, which are different from that for conventional SRAM-based FPGAs. If an application circuit that includes some multiplexers (MUXs), address counters, and pipeline registers for control logic is implemented on a VS-FPGA with a conventional technology mapping algorithm, the number of required LBs becomes large, for example, 198 LBs for 4 AB. The mismatch between the ratio of AB and LB in the target application circuit and that in the CLB array results in a low area efficiency of the VS-FPGA system. In addition, the MUXs increase not only the number of LBs but also fan-out, which results in poor performance. In contrast, since conventional FPGAs have a lot of LBs, and repeaters in routing resource, utilizing LBs to implement small adders, shift registers, and MUXs is acceptable.

In this paper, we propose a technology mapping algorithm for implementing application circuits on VS-FPGA. This algorithm first analyzes a target circuit and detects MUXs, adders/subtractors, and pipeline registers which can be mapped to AB. Some of these elements may be merged and divided for the alignment to the word width. These elements are evaluated with an index of how resource utilization and fan-out change when the corresponding element is implemented with AB. The elements to be assigned to AB are sorted in descending order based on the index. According to this order, each element is mapped to AB one by one, and synthesis and evaluation are repeated iteratively until satisfying given design constraints. The experimental results show that the number of CLBs and maximum fan-out can be reduced by about 30% to 50% and 12% to 87%, respectively.

II. Via-Switch FPGA Overview

A. Structure and function of via-switch in FPGA

Figure 2 illustrates the structure and function of via-switch. A via-switch consists of two atom switches in series and two varistors [5]. Each atom switch is a non-volatile and rewritable solid-electrolyte switch, and it is composed of a solid-electrolyte sandwiched between Cu and Ru electrodes [6]. By applying a positive or negative voltage, a Cu bridge can be formed or removed (Fig. 2 (a)). By changing the state of both atom switches through the control lines, the left and right signal lines can be conducted or interrupted (Fig. 2 (b)). A detailed programming procedure is out of the scope of this paper and omitted. A two-dimensional via-switch array provides a function of routing crossbar. The via-switch crossbar can also work as a truth table and can be combined with multiplexers to implement a LUT. The LUT in Fig. 3 is 0/1/A/$\overline{A}$-type LUT dedicated for VS-FPGA (for details, see reference [7]). The configuration memory for AB can be implemented in the same way.

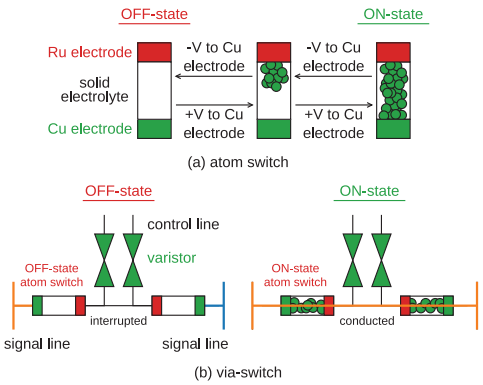

Fig. 2. Structure and function of via-switch.

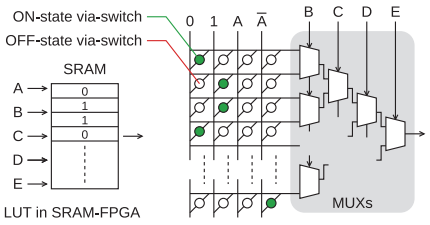

Fig. 3. LUT in VS-FPGA (0/1/A/$\overline{A}$-type LUT) [7].

B. Via-Switch FPGA Architecture

As shown in Fig. 1, VS-FPGA is designed as an array of CLBs, and Arith-CLB and SRAM-CLB are uniformly tiled to achieve efficient local data movement between memories and arithmetic units. The configuration memory, the truth table, and the XB which are responsible for programmability are implemented on BEOL layers as via-switch arrays. On the other hand, the arithmetic block (AB), SRAM, MUXs in LB, and pipeline registers in AB and LB (omitted in Fig. 1) which are responsible for computing, and repeater are implemented on FEOL layers. They are colored red and green in Fig. 1, respectively. This structure allows utilizing FEOL fully for computing. In the case of conventional SRAM-based FPGAs, components for programmability, such as pass-transistor, MUX, and configuration memory are implemented on not only BEOL but also FEOL layers. This reduces the amount of computing resources per unit area, that is, area efficiency of FPGAs.

In the implementation of VS-FPGA [4], architecture parameters such as the size of XB, a number of the LBs in one CLB, size of LUTs, and the architecture of AB have been explored to balance between BEOL and FEOL area in one CLB and between AB-CLB and SRAM-CLB area. The size of Arith-CLB and SRAM-CLB are 118.8um×140.4um and 199.8um×140.4um, respectively. The number ratio of AB, SRAM, and LB in the CLB array is 1:2:8. Each LB contains two 4-input LUTs, which can also operate as a single 5-input LUT. The estimated computation density and energy-efficiency are 29x and 5x higher than MUX-based FPGA [4].

Figure 4 is the block diagram of AB in the VS-FPGA [4]. This AB is composed of adders/subtractors, a multiplier, a bar-

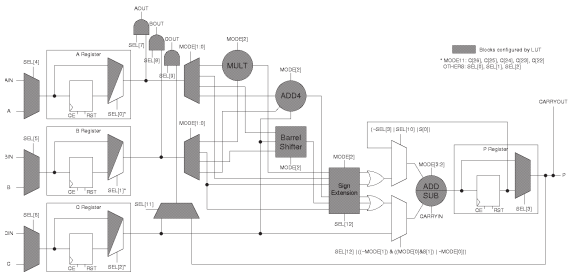

Fig. 4. Block diagram of the arithmetic block (AB) [4].

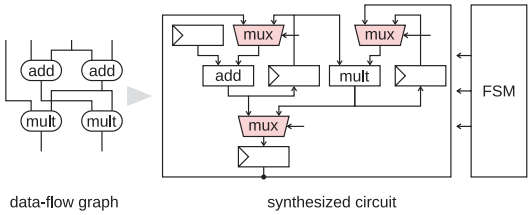

Fig. 5. MUXs for resource sharing generated by high-level synthesis.

rel shifter, and pipeline registers with enable/reset lines, and can perform fundamental arithmetic operations, arithmetic/logical shift operations, and multiply-add/multiply-accumulate operations. This design is intended to enable AB to implement not only arithmetic operations but also control logic such as MUX. The reason for this design is explained in the following subsection.

C. Discussions on VS-FPGA Architecture and Technology Mapping

As described in Section I, we focus on the two features of the VS-FPGA that are different from conventional SRAM-based FPGAs. The first is the small number of repeaters provided in the routing resource. As shown in Fig. 1, most of the wires that connect two XBs in one CLB are directly connected, but repeaters are provided between a few wires. If large numbers of repeaters are implemented in the FEOL layer as well as the computing resource, which reduces the computing area efficiency of the VS-FPGA. On the other hand, a long-distance and/or high fan-out signal path which passes through a large number of repeaters suffers from propagation delay cased by parasitic capacitance. This is because the signal travels back and forth between the FEOL and BEOL layers many times. Therefore, the number of repeaters in the routing resource and its utilization should be minimized to improve the area-efficiency of VS-FPGA. As described in the next section, a select signal for MUXs tends to be high fan-out when MUXs are implemented with LBs. In addition, large-scale application circuits generated by high-level synthesis (HLS) tend to include a lot of MUXs for resource sharing, as illustrated in Fig. 5. A technology mapping that can implement many MUXs efficiently is required not only for the VS-FPGA but also commercial SRAM-based FPGAs.

The second feature is the small ratio of LB to AB. Conventional SRAM-based FPGAs have relatively more LBs than ABs. For example, the ratio of the number of logic cells to DSP slices in Spartan-6/Virtex UltraScale+, which are Xilinx's small/large FPGA, is 768:1 and 308:1 [8], respectively. In contrast, the ratio of the number of LBs to ABs is 8:1 in the VS-FPGA designed with 65-nm CMOS technology [4]. Note that the AB and LB in VS-FPGA correspond to the DSP slice and logic cell in SRAM-based FPGA, respectively. This is a reasonable design to balance the BEOL area and the FEOL area in one CLB. The ratio is expected to be similar for different technology nodes. Therefore, coarse-granularity (word-wise) circuit such as MUX, address counter, and pipeline register which are implemented utilizing logic cells in conventional SRAM-based FPGA, should be implemented with AB in the VS-FPGA. The AB shown in Fig. 4 is designed to implement these functions.

III. Technology Mapping Algorithm for Via-Switch FPGA

A. Preliminary Example

Figure 6 is a typical example of a control circuit that is sometimes required for compute-intensive applications, such as tensor multiplication. In this circuit, one of the results of two 9-bit adders is selected by a MUX based on an output of random logic, and provided as read/write addresses to some memories via the pipeline registers. Logic synthesis and optimization targetted for 5-input LUTs using Yosys [9] report that the required number of LBs to implement this circuit (without the random logic) is 48, that is, the required number of CLB and size of array are 12 ($= \lceil 48/4 \rceil$) and 4×3, respectively. It is also reported that the output of the random logic is connected to the inputs of 9 LBs, that is, the maximum fan-out of the synthesized circuit is 9.

This circuit can be implemented with only four ABs of Fig. 4, as depicted in Fig. 6. The AB can be configured to a 9-bit adder, a 9-bit 2-to-1 MUX, or simple 9-bit register (by using "P Register" only in Fig. 4). One AB can also be configured as a multi-level pipeline register by feedbacking of its outputs, as illustrated in Fig. 9. The required array size is 2×3, which is smaller than that of the LB-based implementation. Note that only CLBs in the odd-columns contain the AB, as shown in Fig. 1. The maximum fan-out is 1 because each signal line, including the output of the random logic, is connected to one of input port of the ABs.

On the other hand, elements to be mapped to ABs should be carefully selected in order to minimize the required number of CLBs and maximum fan-out. For example, if a large number of small-bit-width adders (*e.g.*, 2-bit adders) are mapped to AB, the required number of CLBs will be much larger than when they are mapped to LBs. In the next subsection, the proposed algorithm, which selects appropriate elements to be assigned to the AB in order to improve area efficiency and performance, is explained.

B. Proposed Algorithm

The entire algorithm flow is shown in Fig. 7. The input of the proposed algorithm is an application circuit described in

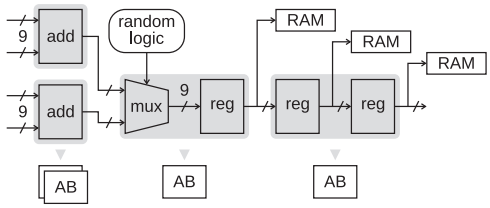

Fig. 6. A typical example of a control circuit.

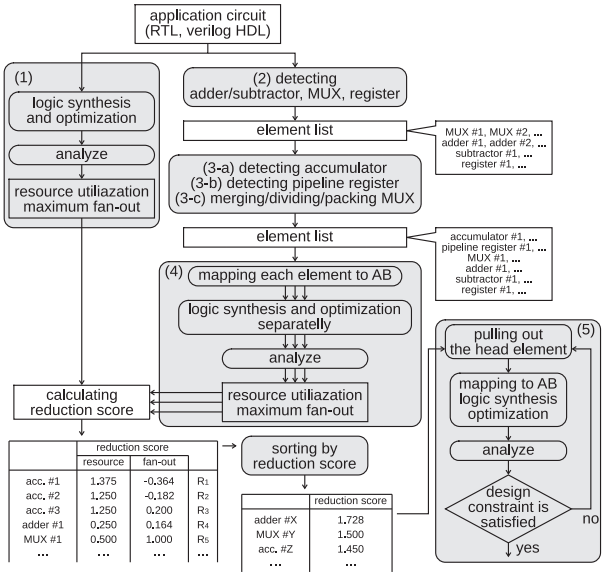

Fig. 7. Entire flow of the proposed algorithm.

Verilog HDL. Note that some complex word-wise functions, such as multiply-add and multiply-accumulate, are manually mapped to AB in advance. First, logic synthesis and optimization (Fig. 7 (1)) for the target circuit are performed to count the numbers of LBs (L_0) and ABs (A_0) for reference. A list of fan-out of all elements ($\boldsymbol{F_0}$) is also created. For example, when the optimized circuit is composed of LBs with 5, 4, 3 and 2 fan-outs, $\boldsymbol{F_0} = (5, 4, 3, 2)$.

On the other hand, adders/subtractors, MUXs, and registers in the target circuit are detected, and a list of these elements is created (Fig. 7 (2)). Three additional processes (3-a, b, c in Fig. 7) are performed to further improve the efficiency of AB utilization, and the element list is updated. Accumulators in Fig. 8 and multi-level pipeline registers in Fig. 9 are detected. Owing to the reset and enable lines in the registers in the AB of the VS-FPGA, the accumulator can be implemented in one AB, as shown in Fig. 8. The details of the algorithm of merging, dividing, and packing MUXs (Fig. 7 (3-c)) are described below.

In order to create a sorted list of elements for prioritizing the mapping to ABs, logic synthesis and optimization are performed separately assuming each element is mapped to AB (Fig. 7 (4)). The numbers of LBs ($L_{1,2,...}$) and ABs ($A_{1,2,...}$) are counted, and lists of fan-out ($\boldsymbol{F_{1,2,...}}$) are created for each optimized circuit. The *reduction score* for each

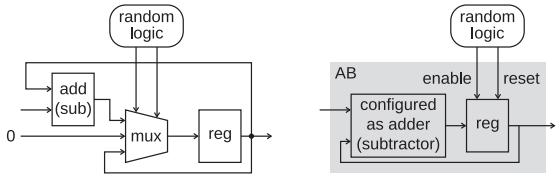

Fig. 8. Example of an accumulator which can be implemented with one AB.

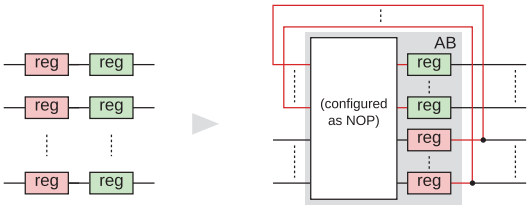

Fig. 9. The AB can be configured as a multi-level pipeline register.

element in the list is calculated. This score is an index of how resource utilization and fan-out change when the corresponding element is implemented with AB. The formula for calculating this score is described below.

The elements in the list are sorted in descending order based on the *reduction score*. Until meeting design constraints such as resource utilization and maximum fan-out, (1) the elements to be mapped to ABs are increased one by one from the head of the sorted list, (2) logic synthesis and optimization are performed for whole application circuit, and (3) the resource utilization and fan-out are analyzed (Fig. 7 (5)).

1) Merging, Dividing, and Packing MUXs: Some MUXs with the same selection signal can be integrated into one MUX and mapped to one AB. The following is the merging, dividing, and packing algorithm to reduce the number of ABs for implementing a lot of MUXs. Some MUXs with a common selection signal are gathered into a group (Fig. 10 (a)). Some large MUXs exceeding the word width of the target ABs are divided into several MUXs (Fig. 10 (b)). Each group is divided into multiple subgroups by a bin-packing algorithm (Fig. 10 (c)), which regards the word width of MUXs and the AB as volumes of items and the fixed volume of bins, respectively.

2) Reduction Score: The reduction score of the i-th element in the list is defined as $R_i = w_r R_{r,i} + w_f R_{f,i}$, where $R_{r,i}$ and $R_{f,i}$ are *resource reduction score* and *fan-out reduction score* of the i-th element, respectively, while w_r and w_f are weights for each score, which are fixed throughout the mapping of an application circuit. Note that the element index i is omitted in the following explanation for readability. The *resource reduction score* is defined as follows,

$$R_r = \frac{L_0 - L_n - N_{\mathrm{LB}}}{(A_n - A_0)N_{\mathrm{LB}}},$$

where N_{LB} is a ratio of the number of LBs to ABs in the target FPGA, that is, $N_{\mathrm{LB}} = 8$ for the VS-FPGA explained in Section II.

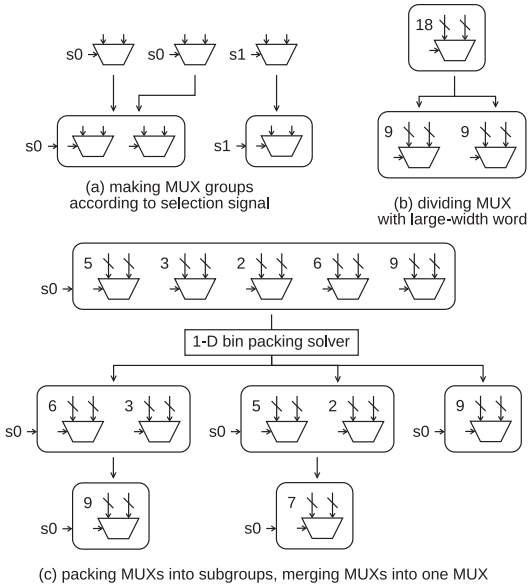

Fig. 10. Algorithm to merge, divide, and pack MUXs. In this figure, the word width of the AB is assumed to 9-bits.

This score gives the reduction of total number of CLBs when the corresponding element is mapped to AB. For example, R_r is 0 (1) when the number of LBs is reduced by N_{LB} ($2N_{\mathrm{LB}}$) by mapping an element to one AB. Conversely, if the number of CLB increases by implementing a corresponding element with AB, R_r becomes a negative value.

The *fan-out reduction score* is defined as follows,

$$R_f = \begin{cases} 1.0 & (\max(\boldsymbol{F_n}) < \max(\boldsymbol{F_0})) \\ {f_n}'/\max(\boldsymbol{F_0}) & ({f_n}' < {f_0}') \\ -{f_0}'/\max(\boldsymbol{F_0}) & ({f_0}' < {f_n}') \\ -\max(\boldsymbol{F_n})/\max(\boldsymbol{F_0}) & (\max(\boldsymbol{F_0}) < \max(\boldsymbol{F_n})) \\ 0 & (\text{otherwise}) \end{cases}$$

where $\max(\boldsymbol{F})$ is the largest value in $\boldsymbol{F}$, and ${f_0}'$ (${f_n}'$) is the largest value in $\boldsymbol{F_0}$ ($\boldsymbol{F_n}$) that does not exist in $\boldsymbol{F_n}$ ($\boldsymbol{F_0}$). When the maximum fan-out is decreased by implementing a corresponding element with AB, R_f becomes 1. If a large (but not maximum) fan-out is removed with mapping a corresponding element to AB, R_f becomes a positive value less than 1. In other cases, R_f becomes a negative value normalized by $\max(\boldsymbol{F_0})$ or zero.

IV. Evaluation

A. Experimental Setup

In this section, we demonstrate that the number of CLBs and maximum fan-out can be reduced by selectively mapping word-wise operations, such as MUX and adders, in the application circuit to AB using the proposed algorithm. The target is the VS-FPGA explained in Section II. The logic synthesis and optimization, and pre-synthesis are performed with Yosys [9] and 5-input LUT. The weight values w_r and w_f are manually configured for each application circuit. The iteration loop of Fig. 7 (5) is terminated when both the number of CLB and maximum fan-out become larger than those of the previous iteration, and the result of the previous iteration is adopted.

As benchmark application circuits, a tensor multiply-accumulate (tensor MAC) and a 512-points FFT are used, which are typical examples of compute-intensive circuits. The architecture of the tensor MAC circuit is similar to the matrix multiply units and accumulators in Tensor Processing Unit (TPU) by Google [10]. This circuit has two parameters W and H and is composed of WH multiply-accumulate units and W accumulate units. These units are manually mapped to AB, while the technology mapping of the other parts is performed with the proposed algorithm. The butterfly unit in the FFT is manually mapped to ABs, and technology mapping of the control circuits, such as memory address calculation, is performed with the the proposed algorithm. In addition, some benchmark circuits in VTR (Verilog-to-Routing) [11] are used to demonstrate the generality of the proposed algorithm. When these circuits are synthesized targetting Xilinx FPGAs, the synthesized circuits will not include a DSP.

B. Evaluation Results

Tables I, II, and III summarize the required numbers of ABs, LBs, and CLBs and the maximum fan-out of each benchmark. The *reference* rows in these tables are the results of logic synthesis and optimization without any technology mapping. The *proposed* rows are the results of logic synthesis and optimization after applying the proposed technology mapping algorithm. The *full-mapped* rows are the results with mapping all the detected elements to the ABs. The numbers in parentheses in LB columns are the ratio of the number of LBs to AB (N'_{LB}). The closer these values are to $N_{\mathrm{LB}} = 8$, the more efficiently the resources in the CLB array are utilized.

In most circuits, N'_{LB} is close to 8 due to the proposed technology mapping, that is, the numbers of CLB in *proposed* are smaller than those of *reference*, and the reduction rate is about 30% to 50%. On the other hand, the reduction rate of the maximum fan-out is about 12% to 87%. In some circuits, *full-mapped* has the lowest maximum fan-out, but the difference from that of *proposed* is small.

The reason why the number of CLBs in the tensor MAC with $(W, H) = (8, 8)$ becomes large with the proposed algorithm is that the control circuit is very small compared to the arithmetic circuit. Even in this case, however, the maximum fan-out has been reduced by as much as 68%, so it can be said that the proposed method was effective from the viewpoint of fan-out reduction, that is, performance improvement.

From these results, it can be said that the proposed technology mapping algorithm is effective in both reducing the number of CLBs and the maximum fan-out. Therefore, it can be expected that the proposed algorithm will contribute to the further improvement of the computation density and energy-efficiency of the VS-FPGA.

C. Discussion

The reduction score in the proposed algorithm is an expected value for improving resource utilization and fan-out when one

TABLE I
RESOURCE UTILIZATION AND MAXIMUM FAN-OUT (TENSOR MAC).

(W, H)	design	AB	LB	CLB	maximum fan-out
$(2, 2)$	*reference*	4	198 (49.50)	50	55
	proposed	14	100 (7.14)	28	11
	full-mapped	32	23 (0.72)	64	15
$(4, 4)$	*reference*	16	363 (22.69)	91	109
	proposed	33	210 (6.36)	66	21
	full-mapped	56	24 (0.43)	112	20
$(8, 8)$	*reference*	64	690 (10.78)	173	217
	proposed	96	383 (3.99)	192	69
	full-mapped	129	26 (0.20)	258	68

TABLE II
RESOURCE UTILIZATION AND MAXIMUM FAN-OUT (512-POINTS FFT).

design	AB	LB	CLB	maximum fan-out
reference	8	315 (39.38)	79	47
proposed	21	167 (7.95)	42	28
full-mapped	52	57 (1.10)	104	16

element is implemented with AB. However, the reduction score when multiple elements are implemented with ABs at the same time is not the sum of their reduction scores. By taking this nonlinearity into account in the calculation of the reduction score, we can further improve the proposed method.

The run-time to calculate the reduction scores (Fig. 7 (4)) is determined by run-time for logic synthesis/optimization and number of detected elements that can be mapped to AB. For example, the run-time of the tensor MAC (whose W and H are 2) and blob_merge are 13 and 25919 seconds, respectively. In order to improve the feasibility of the proposed method and make it easier to apply to large-scale application circuits, it is necessary to develop a reduction score calculation method that is not based on the logic synthesis results.

The proposed algorithm is not limited to the VS-FPGA. For example, the DSP slice of Xilinx's FPGA also has a reset and enable lines of registers, so the accumulator can be implemented in one DSP in the same way. A feasibility study on whether the proposed algorithm can be applied to conventional commercial FPGAs dedicated to compute-intensive applications is future work.

The numbers of ABs are larger than those of LBs in the *full-mapped* tensor MAC and FFT. The proposed algorithm reduces the number of LBs by more than half in the VTR benchmarks. These results suggest that most of the components of some application circuits are MUXs, pipeline registers, accumulators in addition to arithmetic operations. Therefore, processing units dedicated to the efficient implementation of these elements could greatly improve the energy efficiency of FPGAs.

V. CONCLUSIONS

This paper proposed a technology mapping algorithm for implementing application circuits on VS-FPGA. This algorithm detects word-wise functions such as MUXs and accumulators in target circuits, and some of them are implemented with ABs to reduce the number of CLBs and maximum fan-out. The evaluation showed that the number of CLBs and maximum fan-out could be reduced by about 30% to 50% and 12% to 87%, respectively. The proposed algorithm is not limited to the VS-FPGA and is expected to improve computation density and energy-efficiency of various FPGAs dedicated to compute-intensive signal processing applications.

TABLE III
RESOURCE UTILIZATION AND MAXIMUM FAN-OUT (VTR BENCHMARK CIRCUITS).

circuit	design	AB	LB	CLB	maximum fan-out
sha	*reference*	0	3062	766	883
	proposed	240	1350 (5.62)	480	488
	full-mapped	380	1273 (3.35)	760	528
stereovision0	*reference*	0	23902	5976	2352
	proposed	1283	5430 (4.23)	2566	309
	full-mapped	1827	1045 (1.75)	3654	294
stereovision3	*reference*	0	277	70	38
	proposed	25	107 (4.28)	50	25
	full-mapped	33	97 (2.94)	66	24
blob_merge	*reference*	0	8002	2001	176
	proposed	254	5464 (21.51)	1366	155
	full-mapped	1064	3431 (3.22)	2128	185

REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[2] M. Lin, A. E. Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, Feb. 2007.

[3] N. Banno, M. Tada, K. Okamoto, N. Iguchi, T. Sakamoto, M. Miyamura, Y. Tsuji, H. Hada, H. Ochi, H. Onodera, M. Hashimoto, and T. Sugibayashi, "A novel two-varistors (a-Si/SiN/a-Si) selected complementary atom switch (2V-1CAS) for nonvolatile crossbar switch with multiple fan-outs," in *Proceedings of IEEE International Electron Devices Meeting (IEDM)*, Dec. 2015, pp. 2.5.1–2.5.4.

[4] M. Hashimoto, X. Bai, N. Banno, M. Tada, T. Sakamoto, J. Yu, R. Doi, Y. Araki, H. Onodera, T. Imagawa, H. Ochi, K. Wakabayashi, Y. Mitsuyama, and T. Sugibayashi, "Via-switch FPGA: 65nm CMOS implementation and architecture extension for AI applications," in *Proceedings of International Solid-State Circuits Conference (ISSCC)*, Feb. 2020.

[5] N. Banno, K. Okamoto, N. Iguchi, H. Ochi, H. Onodera, M. Hashimoto, T. Sugibayashi, T. Sakamoto, and M. Tada, "Low-power crossbar switch with two-varistors selected complementary atom switch (2V-1CAS; via-switch) for nonvolatile FPGA," *IEEE Transactions on Electron Devices*, vol. 66, no. 8, pp. 3331–3336, Aug. 2019.

[6] M. Tada, K. Okamoto, T. Sakamoto, M. Miyamura, N. Banno, and H. Hada, "Polymer solid-electrolyte switch embedded on CMOS for nonvolatile crossbar switch," *IEEE Transactions on Electron Devices*, vol. 58, no. 12, pp. 4398–4405, Dec. 2011.

[7] T. Higashi and H. Ochi, "Area-efficient LUT-like programmable logic using atom switch and its delay-optimal mapping algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100-A, no. 7, pp. 1418–1426, Jul. 2017.

[8] Xilinx, Inc., "FPGAs & 3D ICs," https://www.xilinx.com/products/silicon-devices/fpga.html.

[9] J. Glaser and C. Wolf, "Methodology and example-driven interconnect synthesis for designing heterogeneous coarse-grain reconfigurable architectures," in *Models, Methods, and Tools for Complex Chip Design*, J. Haase, Ed. Springer International Publishing, 2014, pp. 201–221.

[10] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2017, pp. 1–12.

[11] K. E. Murray *et al.*, "VTR 8: High performance CAD and customizable FPGA architecture modelling," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 13, no. 2, pp. 9:1–9:60, Jun. 2020.