

Architecture of a Multi-Context FPGA Using Reconfigurable Context Memory

Weisheng CHONG, Sho Ogata, Masanori HARIYAMA and Michitaka KAMEYAMA
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba-ku, Sendai,
Miyagi, 980-8579, Japan
{cwsheng, ogata, hariyama, michi}@kameyama.ecei.tohoku.ac.jp

Abstract

Dynamically-programmable gate arrays (DPGAs) promise lower-cost implementations than conventional FPGAs since they efficiently reuse limited hardware resources in time. One of typical DPGA architectures is a multi-context one. Multi-context FPGAs (MC-FPGAs) have multiple memory bits per configuration bit forming configuration planes for fast switching between contexts. The additional memory planes cause significant overhead in area and power consumption. To overcome the overhead, a fine-grained reconfigurable architecture called reconfigurable context memory (RCM) is presented based on the fact that there are redundancy and regularity in configuration bits between different contexts. Switch blocks are efficiently implemented by using RCM as context decoders and routing switches. By using the RCM in logic blocks, an adaptive multi-context logic block table is introduced where the size of look-up tables and the number of different configuration planes of look-up tables are adaptively determined at each logic block. Moreover, non-volatile ferroelectric-based functional pass-gates are used as components of the RCM to achieve compactness and low static power. Under a constraint of the same number of contexts, an area of the proposed MC-FPGA is 45% of that of the conventional MC-FPGA. In the functional-pass-gate-based evaluation, the area of the proposed MC-FPGA is reduced to 37% of the conventional MC-FPGA one.

1. Introduction

Dynamically-programmable gate arrays (DPGAs) [1] provide more cost-effective implementations than conventional FPGAs where hardware resources are dedicated to a single context. A DPGA can be sequentially configured as different processors in real time, and efficiently reuse the limited hardware resources in time.

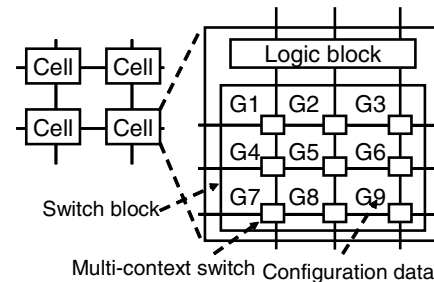


Figure 1. Overall structure of an MC-FPGA

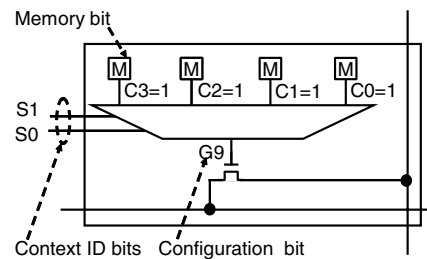


Figure 2. Conventional multi-context switch (four contexts)

One of typical DPGA architectures is a multi-context one. Multi-context FPGAs (MC-FPGAs) have multiple memory bits per configuration bit forming configuration planes for fast switching between contexts. However, the additional memory planes cause significant overhead in area and power consumption [2]. Figure 1 shows the overall structure of an MC-FPGA. Each cell consists of a programmable logic block and a programmable switch block. Figure 2 shows the structure of a conventional multi-context switch. The switch has multiple memory bits for multi-contexts and its contexts are selected from the memory bits according to a context ID. In the con-

	Context 3 (C3)	Context 2 (C2)	Context 1 (C1)	Context 0 (C0)
G1	0	0	0	1
G2	1	0	1	0
G3	0	0	0	0
G4	1	0	1	0
G9	1	1	1	1

Table 1. Redundancy and regularity in configuration data

	Context 3	Context2	Context 1	Context0
S1	1	1	0	0
S0	1	0	1	0

Table 2. Relations between contexts and context-ID bits

ventional approach, each switch requires n bits to store n contexts. Most previous works for DPGAs reduce the overhead using device-level solutions. That is, compact memory devices such as DRAM and FeRAM were used to store configuration data [1, 3].

To reduce the overhead of configuration memory in MC-FPGAs, this paper proposes an architectural-level solution based on the fact that there are redundancy and regularity in configuration bits between contexts. To illustrate the redundancy and regularity, Table 1 shows an example of configuration data of the switch block shown in Fig. 1. Each row denotes configuration data of each switch. The configuration data G3 and G9 have redundancy in themselves. That is, there is no change in their configuration bits. It is said that less than 3% of configuration data are changed when contexts are switched [4]. There is another type of redundancy between configuration data of different switches. For example, G2 and G4 have the same configuration data. Moreover, there is regularity in configuration data such as G2 and G4. The configuration data G2 and G4 can be represented by repeating bits in an order of (0,1). To exploit the redundancy and regularity, a fine-grained reconfigurable architecture called “reconfigurable context memory” is presented. The switch block consists of fine-grained switch elements where each switch element use a single 2-to-1 multiplexer, two memory bits and a pass-gate as components.

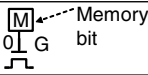

Configuration bit (G)				Hardware generation of G
Context 3 (C3)	Context 2 (C2)	Context 1 (C1)	Context 0 (C0)	
0	0	0	0	
1	1	1	1	

Figure 3. Configuration-bit patterns that are independent of a context ID

The switch elements are used in two ways. Firstly, they are used as programmable interconnections between logic blocks like conventional FPGAs. Secondly, they are used to make reconfigurable decoders that generate configuration bits from the context ID. By exploiting the redundancy and regularity in configuration data, the decoders are configured in an area-efficient way. Moreover, we show that ferroelectric-based functional pass-gates [5] can be used to implement the switch elements for area efficiency. To exploit the redundancy of configuration data in logic blocks for area efficiency, an adaptive multi-context logic block is introduced. The number of inputs of look-up tables (LUTs) and the number of different configuration planes of LUTs are adaptively determined at each logic block. LUTs with a larger number of inputs reduce the total number of required LUTs for a mapping.

Under a constraint of the same number of contexts, the area of the MC-FPGA using the reconfigurable context memory and adaptive multi-context logic block is compared to that of a conventional MC-FPGA. In the CMOS-circuit-based evaluation, the area of the proposed MC-FPGA is 45% of the conventional MC-FPGA one. In the functional-pass-gate-based evaluation, the area of the proposed MC-FPGA is 37% of the conventional MC-FPGA one.

2. Redundancy And Regularity in Configuration Data

Redundancy and regularity in configuration data can be used to reduce the area of the context memory. In this paper, an architecture with four contexts is considered as an example although our approach is also applicable to architectures with other number of contexts. Contexts are switched by a 2-bit context ID (bit S1 and bit S0) as shown in Table 2. For a 4-context switch, there are 16 possible configuration-bit patterns as listed in Figs. 3, 4 and 5. Note that each row in the figures represents one of the configuration-bit patterns for the switch. Figure 3 shows configuration-bit pat-

Configuration bit (G)				Hardware generation of G
Context 3 (C3)	Context 2 (C2)	Context 1 (C1)	Context 0 (C0)	
0	0	1	1	$\overline{S1} \downarrow G$
0	1	0	1	$\overline{S0}$
1	0	1	0	$S0$
1	1	0	0	$S1$

Figure 4. Configuration-bit patterns that depend on a context-ID bit

terns that are independent from the context ID because the switch is programmed to be always turned on or off. A single memory bit is sufficient to control the switch, while four memory bits is required for the conventional switch shown in Fig. 2. Figure 4 shows configuration-bit patterns that depend on a single context-ID bit. Note that each bit pattern is same as the bit patterns of $S1$ (or $\overline{S1}$) or $S0$ (or $\overline{S0}$) shown in Table 2. A switch using a single context-ID bit is smaller than a conventional switch which uses two context-ID bits. Figure 5 shows the other configuration-bit patterns that depend on $S1$ and $S0$. Each bit pattern can be generated using a 2-to-1 multiplexer as shown in the right-most column of Fig. 5. The multiplexer is slightly larger than the hardware to generate the bit patterns shown in Figs. 3 and 4. However, the bit patterns in Fig. 5 are not frequently used in a multi-context architecture since less than 3% of configuration bits change when contexts are switched [4].

3. Switch Block Architecture Using the Reconfigurable Context Memory

Figure 6 shows a basic MC-FPGA architecture that uses reconfigurable context memory (RCM) as switch blocks. A detail architecture will be discussed in the next paragraph.

Figure 7 shows the structure of the RCM that consists of fine-grained switch elements (SEs), programmable switches (denoted by P) and input controllers (denoted by C). A programmable switch connects a vertical track with a horizontal track as shown in Fig. 7(b). An input controller can be programmed to invert its input as shown in Fig. 7(c). An SE consists of a pass-gate, a multiplexer and two memory bits (D1 and D0) as shown in Fig. 8. As described in the previous section, configuration-bit patterns with redundancy and regularity are frequently used. They can be implemented with much simpler circuits than the conventional

Configuration data (G)				Hardware generation of G
Context 3 (C3)	Context 2 (C2)	Context 1 (C1)	Context 0 (C0)	
0	0	0	1	$\overline{S0}$ $\overline{S1} \downarrow \overline{S0}$
0	0	1	0	$S0$ $\overline{S1} \downarrow S0$
0	1	0	0	$\overline{S0}$ $S1 \downarrow \overline{S0}$
0	1	1	0	$S0 \overline{S0}$ $S1 \downarrow S0 \overline{S0}$
0	1	1	1	$\overline{S0}$ $\overline{S1} \downarrow \overline{S0}$
1	0	0	0	$S0$ $S1 \downarrow S0$
1	0	0	1	$\overline{S0} \overline{S0}$ $S1 \downarrow \overline{S0} \overline{S0}$
1	0	1	1	$S0$ $\overline{S1} \downarrow S0$
1	1	0	1	$\overline{S0}$ $S1 \downarrow \overline{S0}$
1	1	1	0	$S0$ $\overline{S1} \downarrow S0$

Figure 5. Configuration-bit patterns that depend on two context-ID bits

circuits as shown in Figs. 3 and 4. The RCM is designed in such a way that the frequently-used configuration-bit patterns are implemented area-efficiently using a single SE. For an example, to implement a switch with a configuration bit-pattern shown in the bottom row of Fig. 3, D0 is 1 and D1 is 0. As another example, to implement a switch with a configuration bit-pattern shown in the bottom row of Fig. 4, D1 is 1 and a multiplexer variable input (U) is connected to $S1$. Configuration-bit patterns shown in Fig. 5 are not frequently used and are implemented using several SEs. Figure 9 shows an example to generate the configura-

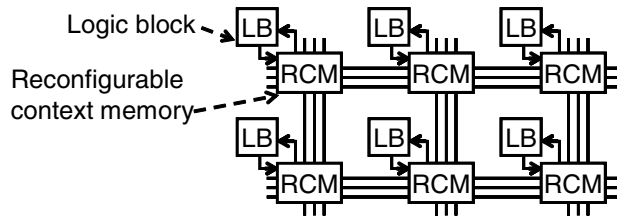


Figure 6. Basic MC-FPGA architecture using the reconfigurable context memory as switch blocks

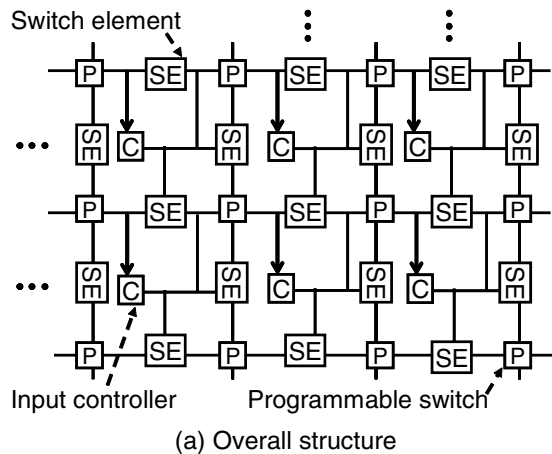


Figure 7. Structure of the reconfigurable context memory

bit patterns where $(C3, C2, C1, C0) = (1, 0, 0, 0)$. Four SEs are sufficient to form the multiplexer. Wires that are used to form the multiplexer are indicated by thick lines.

High speed double-length lines are used in the MC-FPGA to complement routing delay in the RCM. The delay is large if a signal is routed through many SEs in series. Figure 10 shows double-length lines that bypass alternate diamond switches. Each diamond switch connects a line from one direction to another three lines at different directions. The double-length lines are connected to the logic blocks through RCM blocks. Figure 11 shows a diamond switch

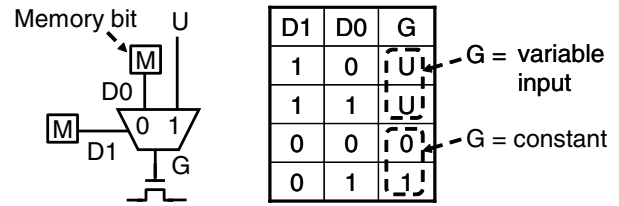


Figure 8. Structure and function of a switch element

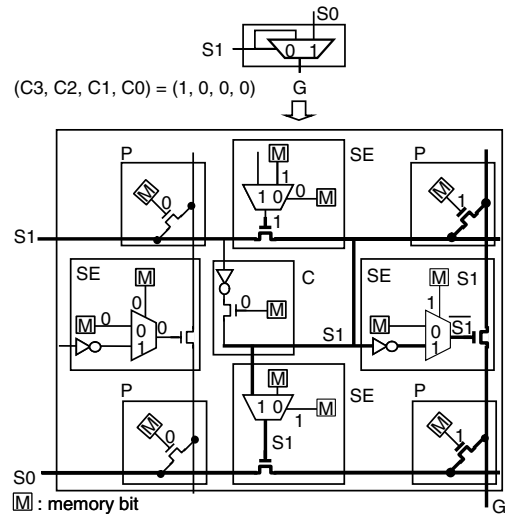


Figure 9. Combination of switch elements to generate the configuration-bit pattern $(C3, C2, C1, C0) = (1, 0, 0, 0)$

that consists of SEs and connects to the RCM through U1 to U6. To achieve a short delay time in a mapping, critical paths are routed with double-length lines while non-critical paths are routed with RCM. To prevent RCM from degrading the context-switching speed, context-ID bits are routed with high-speed global wires and decoded locally with the RCM.

4. Architecture of an Adaptive Multi-Context Logic Block

The main component of an adaptive multi-context logic block is a locally controlled multi-context multi-granularity LUT (MCMG-LUT). Figure 12 shows an MCMG-LUT that is programmable to be a 4-input LUT (four different config-

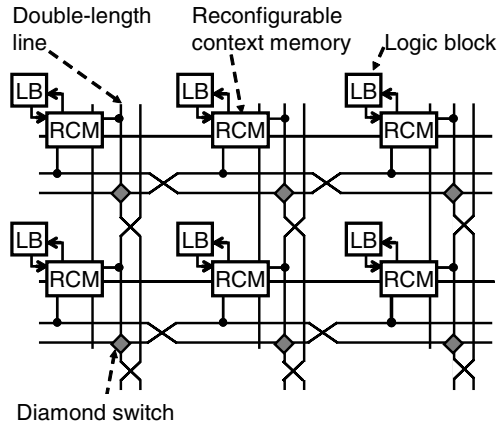


Figure 10. Switch-block structure with double-length lines

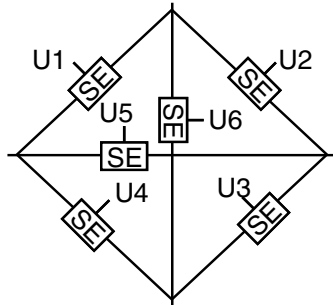
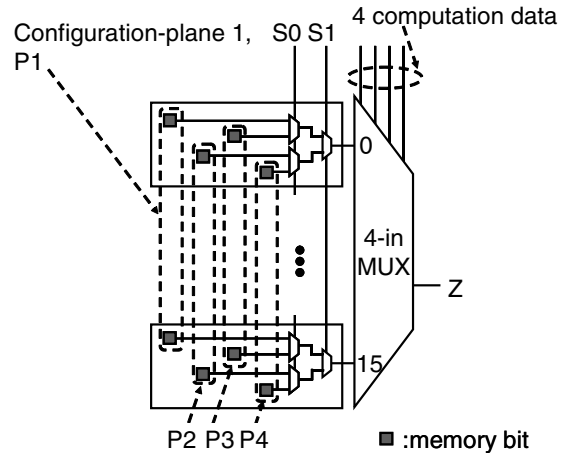


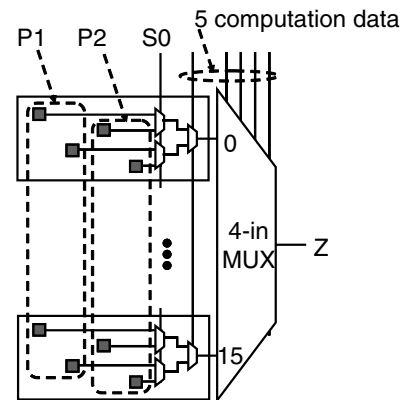
Figure 11. Structure of the diamond switch

uration planes) or a 5-input LUT (two different configuration planes), where each dashed box represents a configuration plane. A configuration plane is a group of memory bits that are selected under the same context-ID state. Note that two configuration bits (S_0, S_1) are used in the 4-input LUT and only one configuration bit (S_0) is used in the 5-input LUT. Without changing the number of memory bits, the size of an MCMG-LUT [6] can be increased by reducing its number of different configuration planes. The size represents the number of computation data that are selected as inputs of an LUT.

Figure 13 shows the mapping of DFGs in contexts 1 and 2 into globally controlled MCMG-LUTs. A global control signal (J) programs each of the MCMG-LUTs as a 2-input LUT (two different configuration planes) as shown in Fig. 13(b). Using a global control signal is not area-efficient because redundant configuration data is stored in the MCMG-LUTs. For example, two configuration planes of LUT3 in



(a) 4-input LUT with four different configuration planes



(b) 5-input LUT with two different configuration planes

Figure 12. Number of different configuration planes and the size for a multi-context multi-granularity LUT

Fig. 13(b) store the same configuration data for O_3 that is repeated in contexts 1 and 2.

Figure 14 shows the mapping of the DFGs into locally controlled MCMG-LUTs to achieve area efficiency. The DFGs are redrawn as shown in Fig. 14(a) where nodes O_2 and O_3 are shared between contexts 1 and 2, and the shared nodes are combined as O_5 . Figure 14(b) shows that two locally controlled MCMG-LUTs are sufficient to map the redrawn DFG compared to three globally controlled MCMG-LUTs shown in Fig. 13(b). Each locally controlled MCMG-LUT has a programmable size-controller that causes area overhead if a dedicated controller is used. To reduce the area overhead, the RCM is used to form the controller that is only required when there are different configuration planes.

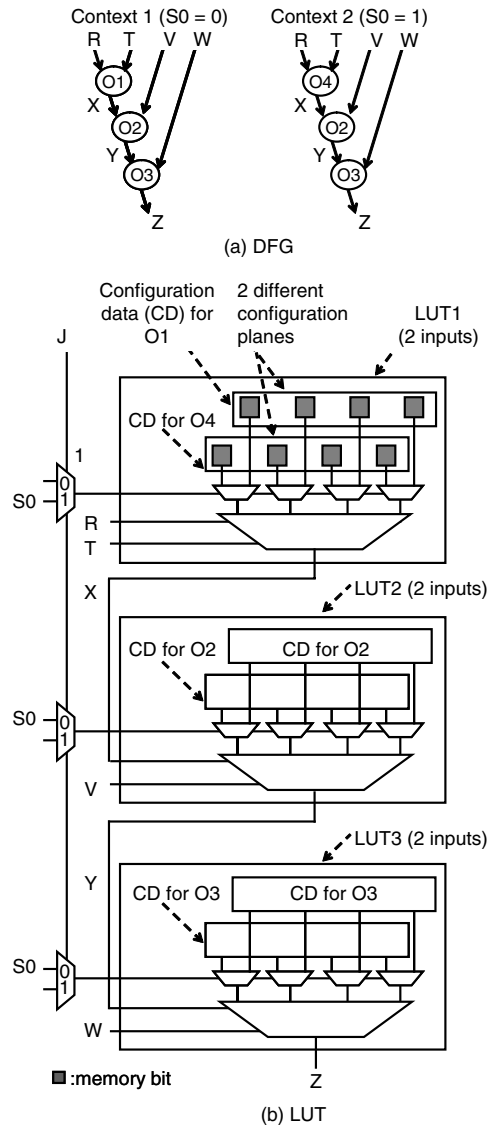


Figure 13. Mapping of DFGs into globally controlled MCMG-LUTs

5. Evaluation

The proposed MC-FPGA using the RCM and adaptive multi-context logic blocks is compared with a typical MC-FPGA. The typical MC-FPGA uses switch blocks and logic blocks with fixed context memory. Let us assume that the number of contexts is four and 6-input 2-output MCMG-LUTs are used. The percentage of changes in configuration data between contexts is assumed to be 5% based on the fact that less than 3% of new configuration memory bits are dif-

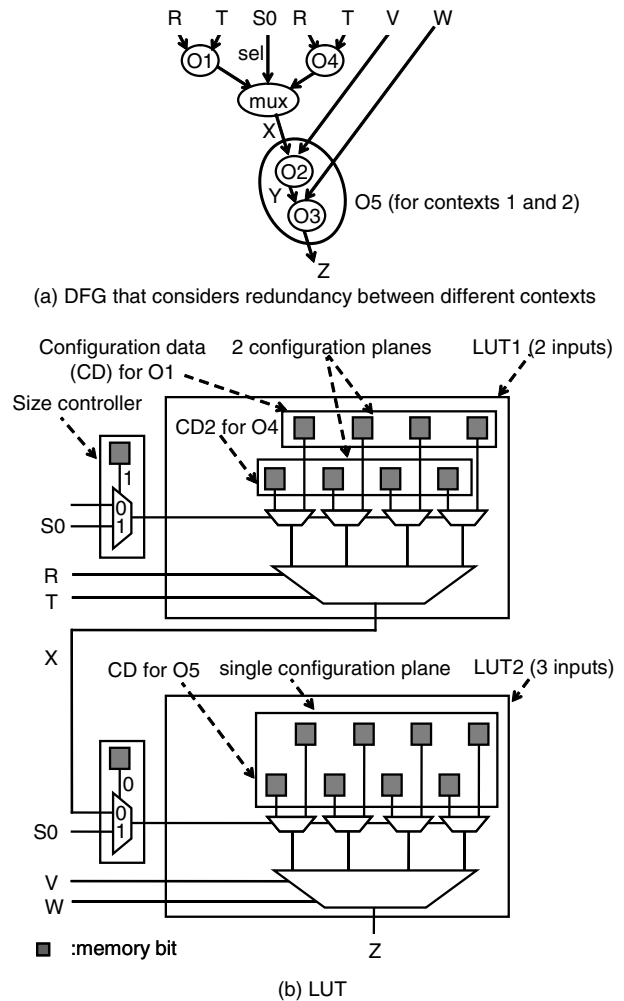


Figure 14. Mapping of a DFG into locally controlled MCMG-LUTs

ferent from those already in the configuration memory [4]. Under a constraint of the same number of contexts, an area of the proposed MC-FPGA is 45% of the area of the typical MC-FPGA.

The RCM can be implemented area-efficiently by using ferroelectric-based functional pass-gates (FePGs) [5] as SEs. FePGs are compact because logic and storage functions are merged at the device level. FePGs can also reduce static power consumption because configuration data are stored in non-volatile ferroelectric devices. Figure 15 shows the circuit of an FePG, its equivalent CMOS circuit and its truth table. Same as an SE, an FePG selects a constant or a variable input depending on contents of two mem-

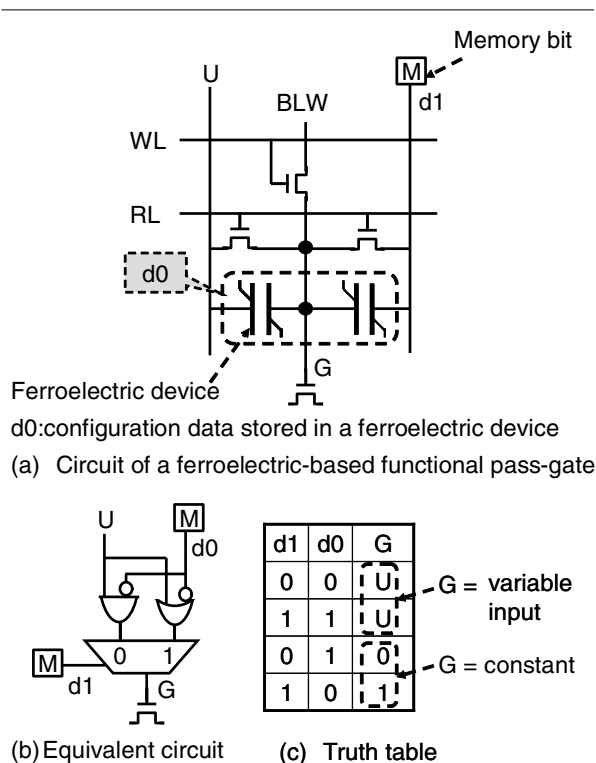


Figure 15. Ferroelectric-based functional-pass-gate

ory bits. The area of an FePG-based SE is 50% of that of a CMOS-based SE. The area of the proposed MC-FPGA using FePG-based SEs is estimated to be 37% of that of a typical CMOS-based MC-FPGA.

6. Conclusion

An MC-FPGA architecture using reconfigurable context memory and adaptive multi-context logic blocks is proposed to reduce overhead of configuration memory. Mapping tools that exploit regularity and redundancy of configuration bits will be investigated in the future to support the architecture.

Acknowledgment

This work was supported in part by Industrial Technology Research Grant Program from New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

- [1] A. DeHon. Dynamically programmable gate arrays: a step toward increased computational density. In *Proceedings of the Fourth Canadian Workshop on Field-Programmable Devices*, pages 47–54, 1996.
- [2] S. Trimberger et al. A time-multiplexed FPGA. In *FCCM'97 Proceedings*, pages 22–28, 1997.
- [3] S. Masui et al. A ferroelectric memory-based secure dynamically programmable gate array. *IEEE J. Solid-State Circuits*, 38(5):715–725, May 2003.
- [4] I. Kennedy. Exploiting redundancy to speedup reconfiguration of an FPGA. In *FPL*, pages 262–171, Sep. 2003.
- [5] H. Kimura, T. Hanyu, M. Kameyama, Y. Fujimori, T. Nakamura, and H. Takasu. Complementary ferroelectric-capacitor logic for low-power logic-in-memory VLSI. *IEEE J. Solid-State Circuits*, 39(6):919–926, June 2004.
- [6] T. M.Iida. A proposal of programmable logic architecture for reconfigurable computing. In *Proc. of ITC-CSCC*, volume 3, pages 1547–1550, 2002.