A Multi-Context FPGA Using a Floating-Gate-MOS Functional Pass-Gate and Its CAD Environment

Masanori Hariyama and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi,980-8579, Japan
Email: {hariyama@, ogata@kameyama., kameyama@}ecei.tohoku.ac.jp

***Abstract*— 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 MC-FPGA architecture using a floating-gate-MOS functional pass gate(FGFP) is presented which merges threshold operation and storage function on a single floating-gate MOS transistor. The test chip is designed using a 0.35μm CMOS-EPROM technology. The transistor count of the proposed multi-context switch (MC-switch) is reduced to 13% in comparison with SRAM-based one. The total area of the proposed MC-FPGA is reduced to about 56% of that of a conventional SRAM-based MC-FPGA.**

***Keywords*—FPGA, DPGA, high-level synthesis, bit-serial architecture**

I. Introduction

Dynamically programmable gate arrays(DPGA) provide more cost-effective implementations than conventional FPGAs where hardware resources are dedicated to a single context. Several different architectures have been proposed [1], [2]. A DPGA is sequentially configured as different processors in real time, and efficiently re-uses the limited hardware resources in time. One typical architecture is a multi-context one. Multicontext FPGAs (MC-FPGAs) have multiple memory bits per configuration bit forming configuration planes for fast switching contexts. However, the additional memory planes cause significant overhead in area and power consumption.

To reduce the overhead, this paper presents following two concepts.

Fine-grained architecture: In conventional FPGAs such as Xilinx XC4000, relatively large logic blocks are used for flexibility. For example, two 4-input LUTs are used in XC4000. The major disadvantage of the large-size logic block is that the logic block is underutilized to a great degree. To solve this problem, fine-grained architecture [3] is used to increase the LUT utilization.

Floating-Gate-MOS functional pass gate: Since the area of the configuration memory is dominant in area, this paper presents a multicontext switch (MC-switch) using a non-volatile functional pass gate. The functional pass gate using a floating-gate MOS transistor(FGMOS) is the device that can merge a logic operation and storage in a single FGMOS[4]. The MC-FPGA is constructed by MC-switches

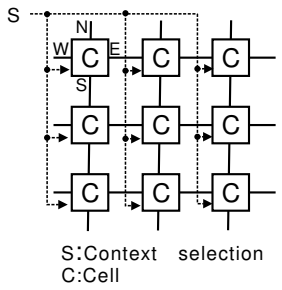

Fig. 1. Overall structure of an MC-FPGA.

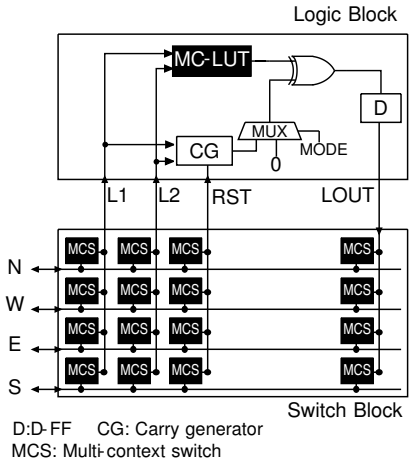

Fig. 2. Structure of a cell.

that can generate an arbitrary ON/OFF pattern according to a context switching signal. The arbitrary ON/OFF pattern can be represented by combining logic functions called "window literals", each of which is efficiently implemented by using the functional pass gates.

II. Fine-grained architecture

As shown in Fig. 1, the FPVLSI consists of a mesh-connected cellular array based on a bit-serial architecture to reduce the complexity of the switch block. Each cell is connected to only four adjacent cells, and the number of programmable switches is reduced in comparison with the typical FPGA. The specification of the functionality of a logic block provides a great impact on the complexity of the switch block, the area of the logic block, and the delay

1-4244-0387-1/06/$20.00 ©2006 IEEE

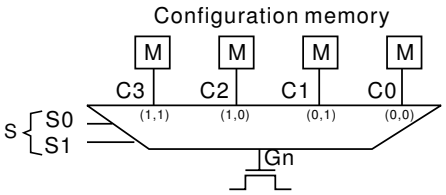

Fig. 3. Equivalent circuit of multi-context switch (4 contexts).

of the logic block. The higher functionality of the logic block requires the larger number of inputs and outputs of a logic block. This increases the complexity of the switch block. On the other hand, the higher functionality of the logic block reduces the number of the logic blocks required for implementing a target function, and reduces the total delay. Based on this observation, the functionalities of the fine-grain logic block are specified as follows:(1) 1-bit addition with carry storage, (2) Arbitrary logic function of two inputs, and (3) 1-bit storage. This specification allows to perform a bit-serial addition on a single cell without carry ripple, and the high functionality can be achieved with the minimum number of inputs and outputs of the logic block. Figure 2 shows the block diagram of the cell. A cell consists of a logic block (LB) and a switch block (SB). The context selection signal CS is commonly distributed to the multi-context LUT(MC-LUT), and multi-context(MC) switch, but it is omitted to be drawn to simplify the figure. The LB consists of an MC-LUT, a carry generator (CG), an XOR gate, a pipeline register (D-FF) and a 2-to-1 multiplexer. Use of the small-size LUT increases the utilization of hardware resource. The CG consists of a 2-in-1 MUX and a D-FF area-efficiently based on the inverting and symmetrical property of the full-adder function[5]. The LB has three modes: logic, delay and arithmetic modes, one of which is selected by a signal MODE. In the case of a single context, the performance of the fine-grained bit-serial architecture is more than two times higher than that of the conventional FPGA architecture under a constraint of the same chip area[3].

Let us explain the MC switch that is the most essential unit in the switch block and the MC-LUT. Figures 3 and 4(a) show an equivalent circuit and an example of the function of an MC switch for 4 contexts, respectively. The context signal $S = (S0, S1)$ is used to select the configuration bit G_n. As shown in Fig. 3, the configuration bits $C0$,$C1$,$C2$ and $C3$ are stored in the configuration memory bits in advance. A configuration memory bit M is programmed to 0 or 1 in advance. One of $C0$, $C1$, $C2$, and $C3$ is selected as the configuration bit corresponding to $(S1, S0)$=$(0, 0)$,$(0, 1)$,$(1, 0)$ and $(0, 0)$, respectively. For the case of N contexts, the MC switch requires N memory bits and the context selection signal with $log_2 N$ bits.

Figure 5 shows the block diagram of the MC-LUT, where the dynamic circuit is used. The input $L1$ and $L2$ are used as input variables of the logic function implemented by the MC-LUT. When the context selection signal is given, the

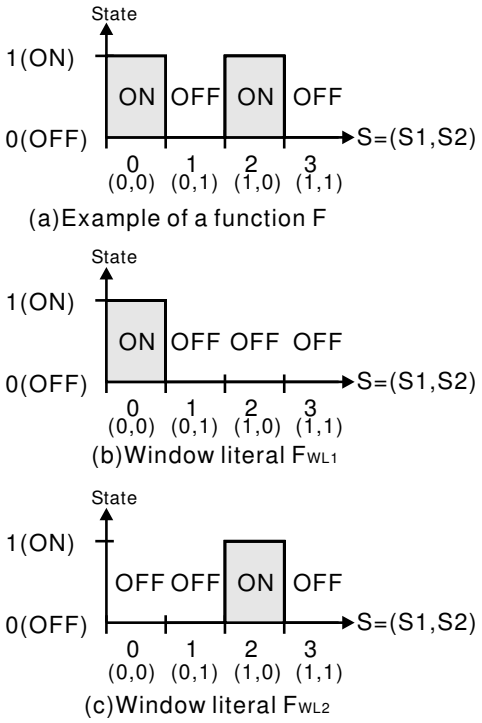

Fig. 4. Function of the multi-context switch (4 contexts).

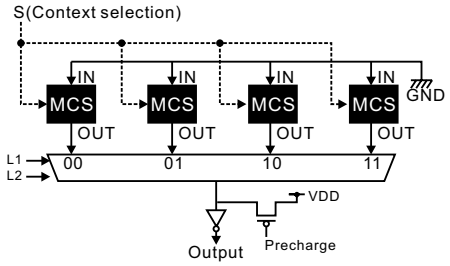

Fig. 5. Block diagram of the MC-LUT.

ON/OFF state of an MC switch is determined, in other words, the configuration bits of an MC switch is determined. The output of the MC-LUT corresponding to $(L1, L2)$ is selected from the configuration bits.

III. Implementation of an MC switch using an FGMOS functional pass gate

Let us consider the function F shown in Fig. 4(a). The function can be given by *OR*-ing the functions F_{WL1} (Fig.4(b)) and F_{WL2}(Fig.4(c)). The functions F_{WL1} and F_{WL2} are window literals. A window literal is defined as

$$F_{WL}(S, S1, S2) = \begin{cases} 1 & S1 \leq S < S2 \\ 0 & otherwise \end{cases} \quad (1)$$

when threshold values $S1$ and $S2$ $(S1 \leq S2)$ are given.

By using Eq.(1), F is given by *OR*-ing 2 window literals as follows:

$$F(S) = F_{WL}(S, 0, 1) + F_{WL}(S, 2, 3) \quad (2)$$

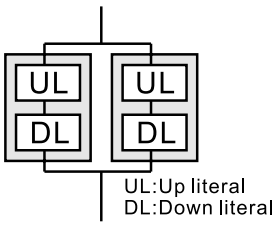

Fig. 6. Circuit of an MC switch (4 contexts).

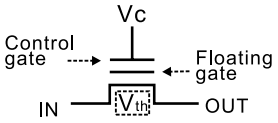

Fig. 7. Floating-Gate MOS transistor.

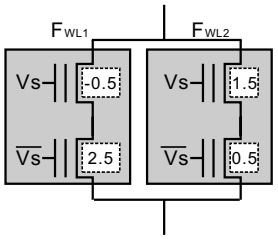

Fig. 8. Implementation of the MC switch using FGFPs.

For the case of N contexts, the function of an MC switch can be given by *OR*-ing $N/2$ window literals at most.

The window literal can be *AND*-ing the functions F_{UL} and F_{DL} that are called "up-literal" and "down-literal" respectively. An up-literal is a monotone increasing function. Given the threshold value T, an up-literal $F_{UL}(S,T)$ is given by

$$F_{UL}(S,T) = \begin{cases} 1 & T \leq S \\ 0 & otherwise \end{cases}$$

A down-literal is a monotone decreasing function. Given the threshold value T, a down-literal $F_{DL}(S,T)$ is given by

$$F_{DL}(S,T) = \begin{cases} 1 & S \leq T \\ 0 & otherwise \end{cases}$$

Hence, the window literal F_{WL2} is expressed as

$$F_{WL}(S,2,3) = F_{UL}(S,2) \cdot FL_{DL}(S,2)$$

Finally, Eq.(2) can be rewritten as

$$\begin{aligned} F(S) \quad = \quad & F_{UL}(S,0) \cdot FL_{DL}(S,0) \\ & + F_{UL}(S,2) \cdot FL_{DL}(S,2) \end{aligned}$$

From this equation, the circuit of the MC switch for N contexts is provided as shown in Fig. 6. The function of the 4-context MC switch is generated by wired-*OR*-ing the outputs of 2 window literals. In general, the function of the N-context MC switch is generated by wired-*OR*-ing the outputs of less than $N/2$ window literals. The output of each window literals is generated by wired-*AND*-ing the outputs of an up-literal and a down-literal. If we use the FGMOS as a 4-valued device, each of an up-literal and a down-literal is implemented by a single FGPG[4] where an FGMOS is used not only as a storage device but also pass transistor. The threshold value of an up-literal or a down-literal is programmed by injecting a controlled amount of electrons into the floating gate. Figure 7 shows the symbol of the FGFP, where threshold voltage is denoted by V_{th} bounded by a broken line. Figure 8 shows the implementation of the switch function F shown in Fig.4, where V_S and $\bar{V_S}$ denotes the control-gate voltages corresponding to S and $\bar{S}$. In N-valued logic, $\bar{S}$ is defined as $N - S - 1$. Hence, $\bar{S} = 3 - S$ for $N = 4$. Note that the down literal F_{DL} for S can be implemented by the up literal for $\bar{S}$.

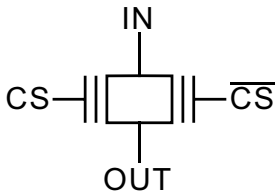

Fig. 9. MC switch for 2 contexts using FGMOSs.

Process	0.35μm CMOS-OTP
Number of contexts	2
Supply voltage	3.3V(6.0V for writing)
Delay of a logic block	1.2ns
Delay of a switch block	2.3ns
Maximum clock frequency	285MHz

TABLE I
FEATURES OF THE MC-FPGA.

IV. IMPLEMENTATION AND EVALUATION

As a first step, the proposed MC-FPGA is implemented for 2 contexts using an FGMOS as a binary device. Figure 6 shows the MC switch for 2 context. Figure 10 shows the micrograph of the MC-FPGA designed using a 0.35μm CMOS-EPROM process of Chartered semiconductor manufacturing. The chip contains 4×4 cells, and circuit for writing test on a $5mm\times5mm$ chip. The supply voltages at the operating and writing modes are 3.3V and 6.0V, respectively. Table I summarizes the features of the proposed MC-FPGA. Table II summarizes the required number of transistors of an MC-switch for 2 contexts. The item "EPROM-based" means an MC-switch using floating-gate MOS transistors just for storage in the equivalent circuit shown in Fig. 3. The item "SRAM-based" means an MC-switch that uses SRAM cells for storage. The number of transistors of the proposed MC-switch is 40% and 13% in comparison with that of the "EPROM-based" and "SRAM-based", respectively. The size of the MC-switch determins the size of a switch block, namely, the wire capacitance of a track. A large wire capacitance causes a large delay and power consumption. Table III compares the proposed switch block and the SRAM-based one in delay and power consumption. The area of the proposed cell is 56% of the SRAM-based one. As shown in Table III, the area and power of the proposed switch is 46% and 31%, respectively.

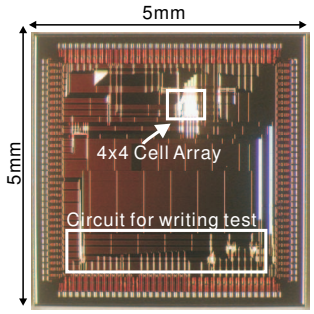

Fig. 10. Micrograph of the proposed MC-FPGA.

Proposed	EPROM-based	SRAM-based
2	5	15

TABLE II

NUMBER OF TRANSISTORS OF AN MC-SWITCH FOR 2 CONTEXTS.

	Proposed	SRAM-based
Delay (Normalized)	0.46	1
Power consumption (Normalized)	0.31	1

TABLE III

COMPARISON OF A SWITCH BLOCK IN DELAY AND POWER CONSUMPTION.

V. CAD ALGORITHM

Our design flow consists of (1) finding correspondence between, (2) combining DFGs, and (3) Place & Route. Behavioral description is given as a data-flow graph (DFG) where a node and an edge between two nodes represent an operation and data dependency. The major differences between conventional CAD algorithms and ours are steps (1) and (2). Step (1) finds the sub-DFGs in different contexts that have the same structure, that is, the same topology. If contexts are mapped to cells independently of each other, it increases complexity of interconnection and the number of used cells. Therefore, sub-DFGs in different contexts should be mapped to same cells if they have the same structure. Figure 11 shows the example. For simplicity, we use DFGs with the same structure as shown in the second row of Fig. 11 (a). If the DFGs are mapped in different ways as shown in the third row of Fig. 11 (a), different interconnections are used as shown in Fig. 11 (b). If the DFGs are mapped in the same ways as shown in the fourth row of Fig. 11 (a), the same interconnection is used as shown in Fig. 11 (c). Thus, this mapping can save more interconnection resources. Step (2) makes a super DFG by combining the given DFGs based on the correspondence. The mapping algorithm should map the sub-DFGs with correspondence in the same way. Moreover, it should ensure routing between the nodes in the other parts of DFGs. To satisfy both requirements uniformly, a single DFG called a super DFG is proposed. By using the super DFG, mapping algorithms for conventional single-context algorithms can be used for the proposed MC-FPGA. Let us show an experimental result for three different contexts: Smoothing with 111 nodes, Edge detection with

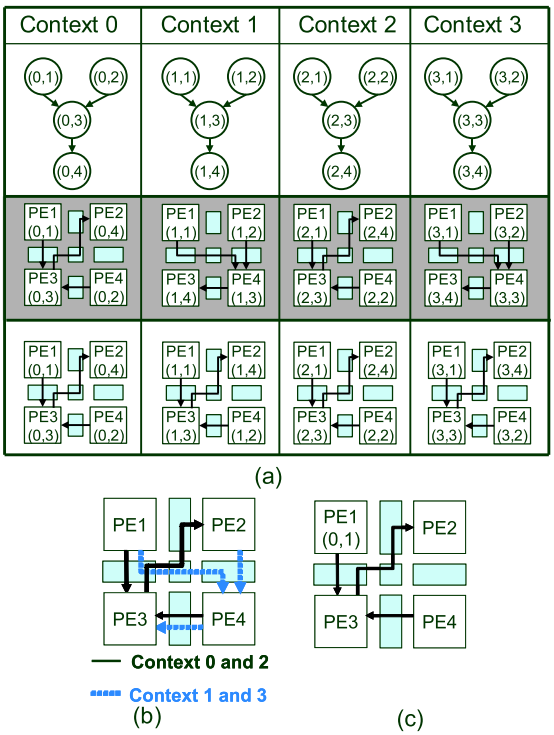

Fig. 11. Correponding contexts.

101 operations, and Template matching with 173 operations. The proposed mapping algorithm found sub DFGs with 95 nodes in the contexts. Therefore, as for smoothing and edge detection, more that 90in the different contexts.

VI. CONCLUSION

This paper presents a novel MC-FPGA architecture. The key technologies are the fine-grained architecture to increase the utilized ratio of LUTs, and an FGMOS functional pass gate to implement an MC switch area-efficiently. As a next step, multiple-valued version is undergoing.

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] N. B. Bhat et al. Performance-oriented fully routable dynamic architecture for a field programmable logic device. In Univ. California, Berkley, CA, Memo. UCB/RELM93/42, 1993.
[2] D. Carberry S. Trimberger and A. Johnson. A time-multiplexed fpga. In IEEE Workshop FPGAs for Custom Computing Machines, pages 22.28, April 1997.
[3] M. Kameyama M. Hariyama, W. Chong. Field-programmable vlsi based on a bit-serial fine-grain architecture. IEICE Trans. Electron., E87-C(11):1897.1902, 11 2004.
[4] M. Kameyama T. Hanyu. Multiple-valued logic-in-memory vlsi architecture based on floating-gate-mos pass-transistor logic. IEICE Trans. Electron., E82-C(9):1662.1668, 9 1999.
[5] Katarzyana Leijten-Nowak. An fpga architecture with enhanced datapath functionality. In FPGA03, pages 195.204, 2003.