Open Source Multi-functional Memory Unit and Application to Approximate Computing

Shigetoshi Nakatake
The University of Kitakyushu, Fukuoka 808-135, Japan
Email: nakatake@kitakyu-u.ac.jp

Approximate computing is one of promising computation techniques which returns a possibly inaccurate result rather than a guaranteed accurate result. Conventionally, this kind of inaccurate computing is allowed for software. However, as growing mobile and embedded devices, the border of software and hardware implementation is no longer strict. We propose a novel multi-functional memory unit which can reconfigure a function of the memory decoder, which is applicable to approximate computing. In our reconfigurable mechanism, uni-switch cells are introduced to play an alternative role of a logic or a wire, and are embedded in an SRAM array. Hence, an extensional function of the decoder is realized by PLA units inside the memory array, and is used for approximate computing. [1] Furthermore, we demonstrate an implementation of our idea on OpenRAM which is an open-source SRAM array compiler.

A) Uni-switch: A typical PLA is composed of AND- and OR-planes, and realizes a logic in the form of the sum-of-product [1]. An example is illustrated in Fig. 1. The most important feature is to realize a multi-input multi-output function, which is suitable for a decoder and an encoder. The programmability of the conventional PLA is implemented by anti-fuses between the nMOS transistors and the wires, but it is like a one-time writing memory. It is preferable that logic and wiring can be reconfigured from external signals by utilizing re-writable memories such as SRAMs like a recent LUT-type FPGA. We introduce a uni-switch to PLA as shown Fig. 2(a). It enables us to change the connection patterns by the external signal A, B and C. The connection patterns are shown in Fig. 2(b).

B) PLA embedded in Memory: In our idea, we superimpose a PLA and an SRAM array. We can configure a logic through the conventional SRAM I/O interface, as well as we can set the input and read the output of the logic circuit via the interface. Combining a uni-switch and three SRAM cells as one programmable cell as shown in Fig. 3, we embedded uni-switches onto a memory array as illustrated in Fig. 4.

Furthermore, all structures of our PLA-based memory unit are generated automatically utilizing OpenRAM introduced in https://openram.soe.ucsc.edu/home, and we demonstrate simulation results for several test data. A generation flow is demonstrated in Fig. 4.

[1]The preliminary of this work is introduced in N. Yahiro, B. Liu, A. Nanri, S. Nakatake, Y. Takashima, G. Chen, *A multi-functional memory unit with PLA-based reconfigurable decoder*, Proc. in ReConFig 2016, DOI: 10.1109/ReConFig.2016.7857145. [2]

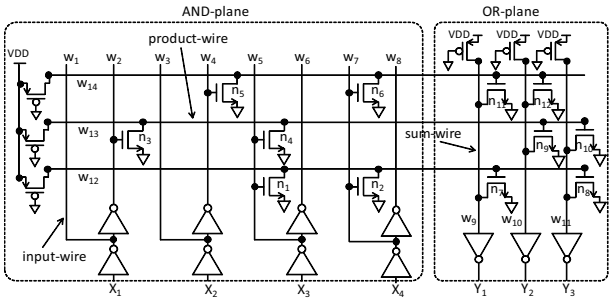

Fig. 1: An example of PLA corresponding to $Y_1 = \bar{X}_2 X_4 + X_3 X_4$, $Y_2 = \bar{X}_2 X_4 + \bar{X}_1 X_3$, and $Y_3 = \bar{X}_1 X_3 + X_3 X_4$.

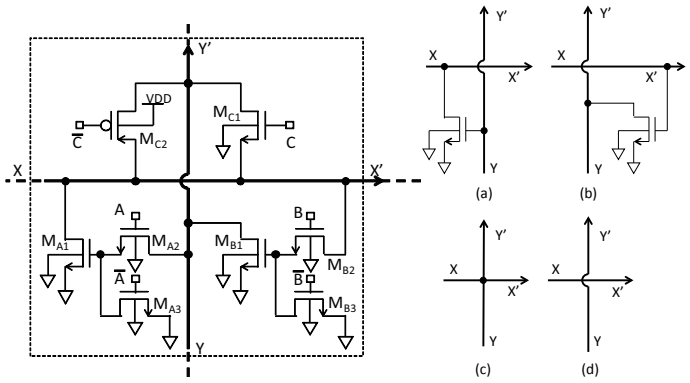

Fig. 2: Uni-switch and four cases of connections.

All structures of the PLA-based memory unit are generated automatically utilizing OpenRAM introduced in https://openram.soe.ucsc.edu/home [3]. Fig. 4 illustrates architecture of memory-array generated by the original OpenRAM. In this environment, we need to prepare a bit-cell and several logic primitives by ourselves. In this work, we extend a bit-cell as containing a uni-switch. As shown in Fig. 3, three bit-cells and a uni-switch are combining into one layout. Note that this is an mp-cell. After generating memory-unit in the original environment, we replace a set of three bit-cells by the mp-cell.

D) Multi-functional Memory: Since our PLA can be embedded in an memory array, the decoder PLA and the memory to store data can be integrated into one SRAM array as shown in Fig. 5. Here, we have a switch matrix outside the SRAM array. The signal 's' switches two modes of the circuit behaviour,

978-1-7281-5020-8/19/$31.00 ©2019 IEEE

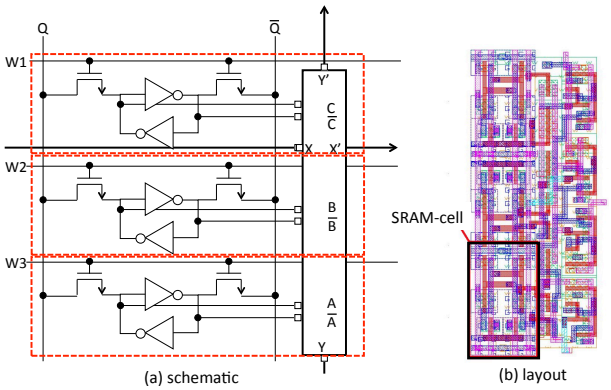

Fig. 3: Memory programmable(mp)-cell combining 3-bit cells.

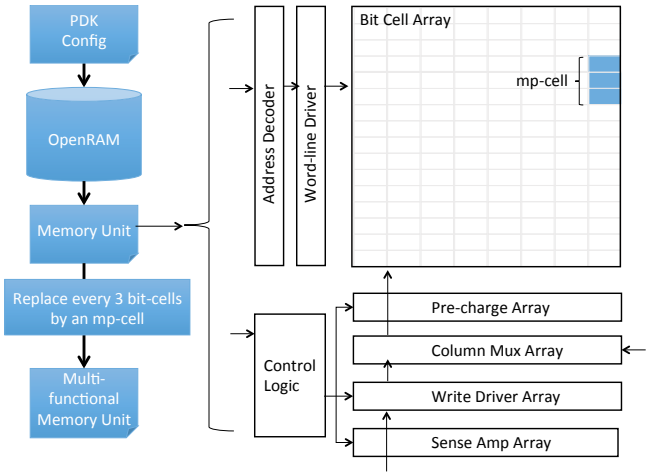

Fig. 4: Memory unit generation with OpenRAM.

normal mode and pre-decoder mode, as follows;
(i) normal mode (s=0): the address data (a_0, a_1, a_2, a_3) is directly mapped to (a'_0, a'_1, a'_2, a'_3) and then the memory array behaves as normal without a pre-decoder.
(ii) pre-decoder mode (s=1): the address data (a_0, a_1, a_2, a_3) is transferred to (x_0, x_1, x_2, x_3) of pre-decoder input. The pre-decoder is realized by PLA. Then, the output of pre-decoder (y_0, y_1, y_2, y_3) is transferred to (a'_0, a'_1, a'_2, a'_3). This means we can realize a function $y_i = f_i(x_0, x_1, \ldots, x_k)$ in the pre-decoder.

E) Approximate Computing: In an application of approximate computing, let a complicated function be $y = f(x)$,where x is in terms of n-bits. The size of table-look-up to store y for each x is 2^n. For saving the size, we introduce a decode function $x' = g(x)$ where $x' < x$, and the decoder function is realized by PLA. Hence, the complicated function is expressed as $y = f(g(x))$.

In this work, for the application to approximate computing, we assume a complicated function demonstrated in Fig. 6(a), which is hard to be represented by a numerical formulation but a set of input and output. Since the input is a floating-point value, we need the floating-point-to-address decoder for realizing this function by our memory-array architecture.

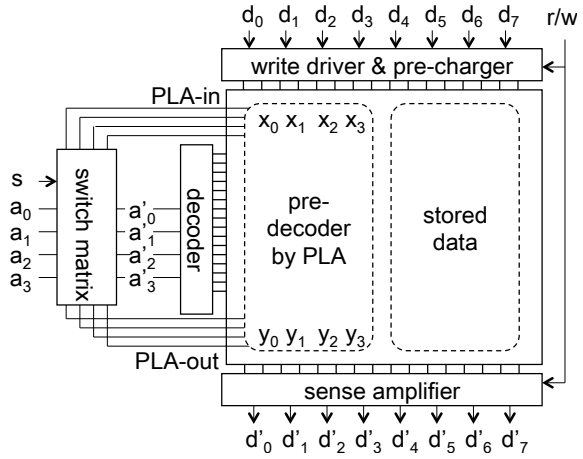

Fig. 5: PLA-based pre-decoder in Memory

Hence, we design a decoder to convert a 6-bit floating-point value to an appropriate memory address, where the exponent is 01, 10, the mantissa is 0000-1111. Hence, the floating-point values 1, 1.0625, 1.125 1.9375, 2.125 3.875 are converted to 32 consecutive addresses. We implement this decoder by PLA corresponding to the truth-table shown in Fig. 6(b), which is a logic of converting a floating-point-to-address. We verify a function implemented on the memory-array by circuit simulation for a netlist combining OpenRAM and PLA-based decoder. The simulation result (Fig. 6(c)) convinces us the function works well. Note that our PLA-based pre-decoder can be reconfigured to various input formats of approximate functions.

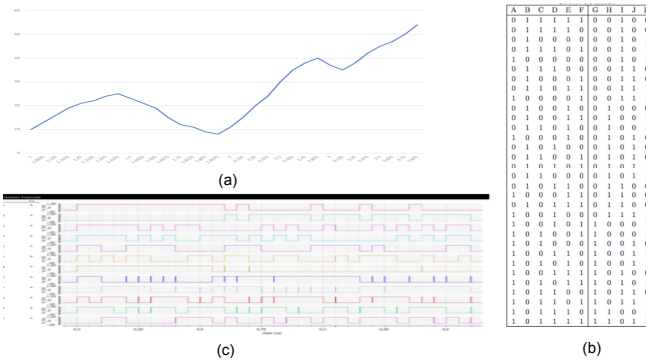

Fig. 6: An example of approximate function: (a) a curve of function, (b) a truth-table of PLA decoder logic, (c) simulation result.

REFERENCES

[1] H. Flesher and L. I. Maissel, An introduction to array logic, IBM Journal of Research and Development, Vol. 19, pp.98–109, 1975.
[2] N. Yahiro, B. Liu, A. Nanri, S. Nakatake, Y. Takashima, G. Chen, A multi-functional memory unit with PLA-based reconfigurable decoder, Proc. in ReConFig 2016, DOI: 10.1109/ReConFig.2016.7857145.
[3] R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, M. Sarwar, OpenRAM: An Open-Source Memory Compiler, Proc. of the 35th International Conference on Computer-Aided Design (ICCAD), 2016 (OpenRAM: https://openram.soe.ucsc.edu/)

978-1-7281-5020-8/19/$31.00 ©2019 IEEE