The 47th IEEE International Midwest
Symposium on Circuits and Systems

Realization of Sequential Circuits by Look-Up Table Rings

Tsutomu SASAO[1] Hiroki NAKAHARA[1] Munehiro MATSUURA[1] Yukihiro IGUCHI[2]

[1] Kyushu Institute of Technology, Iizuka, 820-8502, JAPAN
[2] Meiji University, Kawasaki, 214-8571, JAPAN

Abstract—**The LUT ring is a new type of memory-based realization of a sequential circuit that requires much smaller memory than conventional methods. In this paper, a method to realize a sequential circuit by a look-up table (LUT) ring is presented. The sequential circuit consists of a combinational part and feedback flip-flops. The combinational part is represented by a set of LUT cascades, and they are sequentially emulated by the LUT ring. The LUT ring uses two types of clocks: One evaluates the combinational part, and the other is the clock for state transitions of the sequential circuit. We present a method to reduce the clock period for state transitions, given limited memory size.**

I. Introduction

Two of the most crucial problems in modern VLSI are their long design time and short life cycles. A solution to these problems may be reconfigurable architecture. Reconfigurable architecture will reduce the hardware development time drastically, since one LSI can be used for various applications.

In this paper, we consider a realization of a sequential circuit by reconfigurable architecture. Various methods exist to realize sequential circuits by reconfigurable architecture. Among them, random access memories (RAMs) and programmable logic arrays (PLAs) are easy to design. However, when the number of input variables n is large, the necessary hardware becomes too large.

Design methods for sequential circuits using memories are considered by [3], [4] and [5]. Especially, [5] presents a method to reduce the size of a memory by using multiplexers. It uses a property that most functions depend on proper subsets of the input variables. However, if some output function depends on n variables, then it requires a memory with n-bit address. With $n = 40$, for example, it would be impractical to use the memory with such a size.

This paper shows the realization of sequential circuits by LUT rings. We consider an LUT ring that consists of memory, a programmable interconnection, a control circuit, and some additional circuits. We assume that the target sequential circuit consists of a combinational part and feedback flip-flops. We represent the combinational part by a set of LUT cascades, and sequentially emulate them with the LUT ring. The LUT ring evaluates n variable logic function using a memory that often has less than n address bits. This is possible when the given functions can be decomposed into subfunctions with a smaller number of variables.

The LUT ring is a new type of memory-based realization of a sequential circuit that requires much smaller memory than previous methods.

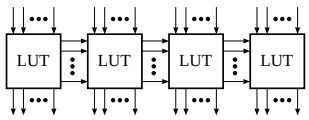

Fig. 1. LUT cascade.

II. Emulation of an LUT Cascade by an LUT Ring

A. LUT Cascades

A combinational LUT cascade realizes a given multiple-output function by the structure shown in Fig.1. The merits of LUT cascades include:

(1) Logic synthesis is relatively easy.
(2) Layout and wiring are very easy.
(3) Delay estimation is easy and accurate.

The demerits of LUT cascades include:

(4) Delay can be larger than random logic networks.
(5) Logic capability is limited once the number of inputs and outputs for each LUT are fixed.

The LUT cascade is obtained from a BDD (Binary Decision Diagram) by iterative functional decompositions [9]. The decomposition algorithm reduces the necessary amount of memory to represent f by detecting repeated patterns in the logic function. The wires connecting adjacent cells in a cascade are called *rails*. In the design of an LUT cascade, the reduction of the number of rails is very important, since it is directly related to the size of the subsequent cell. Let the number of rails be u, and let the width of the BDD be W. Then, we have the relation $u = \lceil \log_2 W \rceil$.

Functions having small BDD widths have efficient LUT cascade realizations, while functions having large BDD widths do not have efficient LUT cascade realizations. For example, adders and symmetric functions have small BDD widths, while multipliers and random functions have large BDD widths. Fortunately many practical functions have BDDs with small widths, and thus have efficient LUT cascade realizations.

B. An LUT Ring that Emulates an LUT Cascade

In an LUT cascade, once the numbers of inputs and outputs for each LUT is fixed, the LUT cascade can realize only a limited class of functions. An LUT ring having

0-7803-8346-X/04/$20.00

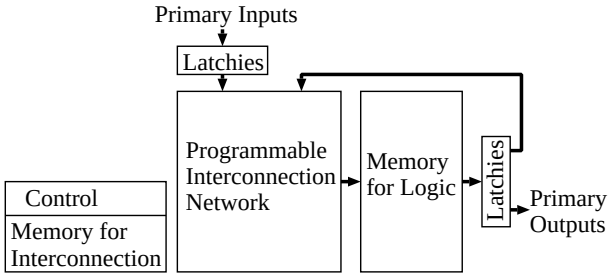

Fig. 2. LUT ring.

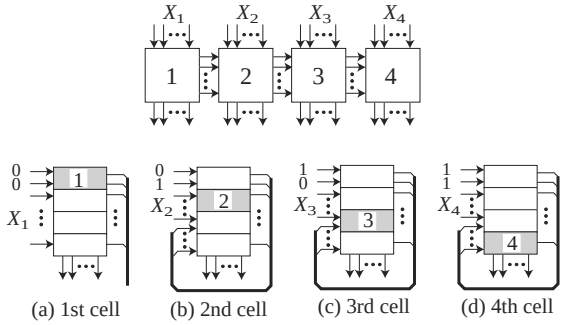

Fig. 3. Operation of an LUT ring.

the architecture shown in Fig. 2 has been developed[1]. It consists of a large memory that stores the data for cells, a programmable interconnection network, and a control part. It sequentially emulates an LUT cascade. Although the LUT ring is slower than the LUT cascade, it has higher logic capabilities than the LUT cascade.

Example 2.1 *Fig. 3 illustrates the emulation of the LUT cascade with four cells by the LUT ring.*

Step 1 *To evaluate the first cell, the two most significant bits of the address are set to (0,0) to specify the 1st page. Also, the values of X_1 are set to the lower address bits through the programmable interconnection network as shown in Fig. 3(a). By reading the contents of the 1st page, the outputs of the 1st cell are computed.*

Step 2 *To evaluate the second cell, the two most significant bits of the address are set to (0,1) to specify the 2nd page. Also, the values of X_2 are set to the middle address bits, and outputs of cell 1 are connected to the least significant bits through the programmable interconnection network, as shown in Fig. 3(b). By reading the contents of the 2nd page, the outputs of the 2nd cell are computed.*

Step 3 *To evaluate the 3rd cell, the two most significant bits of the address are set to (1,0) to specify the 3rd page. Also, the values of X_3 are set to the middle address bits, and the outputs of cell 2 are connected to the least significant bits through the programmable interconnection network, as shown in Fig. 3(c). By reading the contents of the 3rd page, the outputs of the 3rd cell are computed.*

[1]In the previous publication [9], the LUT ring was called the LUT cascade. However, in this paper, the sequential circuit that emulates an LUT cascade is called an LUT ring.

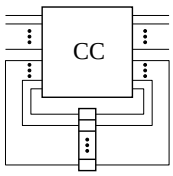

Fig. 4. Model of a sequential circuit.

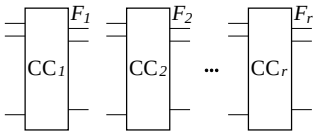

Fig. 5. Partition of output functions.

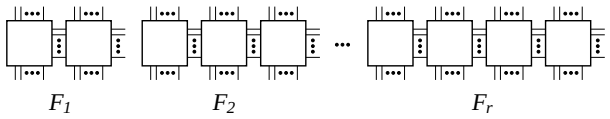

Fig. 6. Cascade realization of groups of functions.

Step 4 *To evaluate the last cell, the two most significant bits of the address are set to (1,1) to specify the last page. Also, the values of X_4 are set to the middle address bits, and the outputs of cell 3 are connected to the least significant bits through the programmable interconnection network, as shown in Fig. 3(d). By reading the contents of the 4th page, the outputs of the last cell are computed.* (*End of Example*)

III. Emulation of a Sequential Circuit by an LUT Ring

A. Emulation of a Sequential Circuit

A sequential circuit can be represented by the model shown in Fig. 4. The combinational part usually has many inputs and outputs, so a direct implementation by a single memory is often impractical. Thus, we partition the outputs of the combinational part into r groups as shown in Fig. 5, and then, realize them by a set of LUT cascade as shown in Fig. 6. Fig. 7 shows the architecture of the LUT ring with a single memory unit. To emulate the sequential machine, the LUT ring uses two types of clock pulses. One is, C_Clock to evaluate each cell of the LUT cascade, and the other is S_Clock for state transitions.

A method to emulate a sequential circuit by the LUT ring is as follows:

1. Partition the outputs into r sets; F_1, F_2, and F_r. Realize each set of outputs by an independent LUT cascade. Let F_i require s_i cells in the LUT cascade, for $i = 1, 2, \ldots, r$. Then, we need $s_1 + s_2 + \cdots + s_r$ cells in total.
2. In evaluating the outputs, for the outputs that becomes primary outputs, store them in the output register, while for the outputs that become state variables, store them in the feedback register.
3. Use double-rank flip-flops shown in Fig. 8 for the feedback register and output register. Set the select sig-

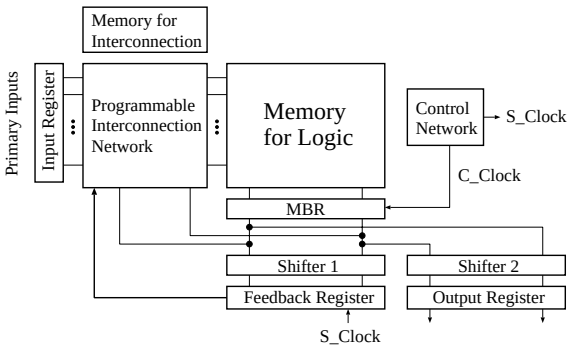

Fig. 7. LUT ring to emulate sequential circuit.

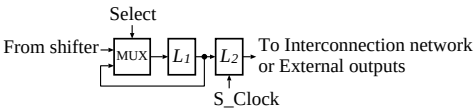

Fig. 8. Double-rank flip-flop.

nals to high when all the cells in a cascade are evaluated, and store the values into the latches L_1.

4. When all the cascades are evaluated, transfer the values of the state variables into the latches L_2. This can be done by adding the pulse to S_Clock. When all the values of the feedback register are ready, they are transfered into the programmable interconnection network. Also, the values of the output register are transfered to the primary outputs.

B. Optimization of an LUT Cascade

In designing an LUT ring, the objects of the optimization are the amount of memory necessary to implement the circuit, and the delay time.

Example 3.2 *Consider the realization of the LUT ring where the combinational part is realized by three LUT cascades as shown in Fig. 9. Note that we need nine memory references to evaluate the combinational part. Let τ be the period of C_Clock, then the period of S_Clock is 9 τ.* (*End of Example*)

The state transitions occur with S_Clock, thus we have the following:

Property 3.1 *The clock cycle of the sequential circuit is proportional to the total number of cells in the LUT cascades.*

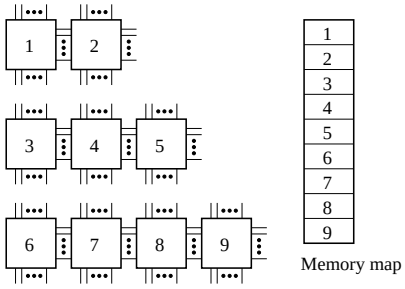

Fig. 9. Realization combinational part by multiple cascades.

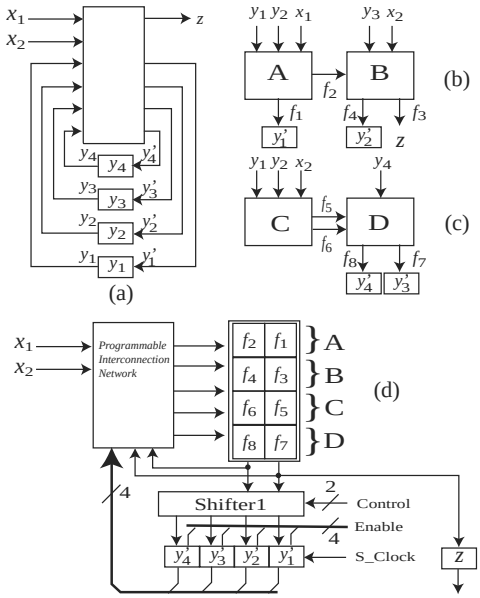

Fig. 10. Operation of sequential circuit.

The optimization problem of the sequential network implemented by an LUT ring is formulated as follows:

Problem 3.1 *Implement the functions by a set of LUT cascades with the minimum number of cells under the given memory restriction.*

Finding an exact optimum solution for this problem is almost impossible, so we will consider a heuristic method:

Algorithm 3.1 *(Realization of a Sequential Circuit)*

1. *Realize an LUT cascade for each output. Let the number of the outputs be m, and let the number of cells of each cascade be s_i $(i = 1, 2, \ldots, m)$. Then, the total number of cells is given by $s_1 + s_2 + \cdots + s_m$.*
2. *Within the memory restriction, reduce the total number of cells in the LUT cascades by merging the outputs.*

To design LUT cascades, we use the BDD that represents the characteristic function of the multiple-output function (BDD for CF) [10].

Example 3.3 *Consider the realization of the sequential circuit shown in Fig. 10(a). It can be emulated as follows: In the first two clock times, f_1, f_2, f_3, and f_4 are evaluated, where $f_1 = y_1'$ and $f_4 = y_2'$ are state variables (Fig. 10(b)). In the next two clock times, f_5, f_6, f_7, and f_8 are evaluated, where $f_7 = y_3'$ and $f_8 = y_4'$ are state variables (Fig. 10(c)). The memory map is shown in Fig. 10(d). Note that the conventional realization requires a 6-input 5-output ROM, while this realization requires a 5-input 2-output ROM.* (*End of Example*)

TABLE I

Realization of sequential circuit by LUT ring.

Name	In	Out	FF	k	Cells	r	Memory
s208	10	1	8	13	2	1	0.187/0.125
s298	3	6	14	13	3	2	0.218/0.187
s344	9	11	15	13	4	2	0.250/0.187
s349	9	11	15	12	4	2	0.097/0.093
s382	3	4	21	12	4	2	0.171/0.156
s386	7	7	6	13	1	1	0.125/0.125
s400	3	6	21	12	4	2	0.171/0.156
s420	18	1	16	14	7	2	0.968/0.906
s444	3	6	21	11	4	2	0.093/0.078
s510	19	7	6	12	3	1	0.187/0.125
s526	3	6	21	14	4	3	0.328/0.281
s641	36	23	19	12	17	3	0.832/0.500
s713	36	23	19	12	22	3	1.257/0.906
s820	18	19	5	14	3	2	0.503/0.503
s1196	13	13	19	14	7	2	0.875/0.750
s1488	8	20	6	12	2	2	0.312/0.312
s1494	8	20	6	14	2	2	0.500/0.500

IV. Experimental Results

We mapped selected MCNC sequential circuits into an LUT ring. Table I shows the results. In this experiment, we assume that each cell may have a different number of inputs. We minimized the total number of cells under the limitation of the 1 Mega bits of memory; we used a memory with 16 inputs and $w = 16$ outputs. In this table, *In* denotes the number of primary inputs; *Out* denotes the number of primary outputs; *FF* denotes the number of flip-flops; k denotes the maximum number of inputs of cells; *Cells* denotes the total number of cells; r denotes the number of cascades; *Memory* denotes the size of memory (mega bits), where the first number shows the size without memory packing and the second number shows the size with memory packing.

For these benchmark functions, we could realize sequential networks by using a memory with at most 1 mega bits. The period of S_Clock is proportional to the number of cells. Thus, the most time-consuming circuit in this table is *s713* that requires 22 cells. The straightforward realization shown in Fig. 4 requires a memory with $36 + 19 = 55$ inputs and $23 + 19 = 42$ outputs. In the LUT ring realization, a 16-input 16-output memory is sufficient, but we need 22 clocks for every state transition and the hardware shown in the next section. However, the LUT ring is much faster than the software simulation on a general microprocessor or branching program machine [7] with the same clock speed.

V. Hardware for LUT Ring

Let w be the number of outputs of MBR (memory buffer register). The LUT ring consists of main memory and the following components: *Programmable Interconnection Network* that selects input signals from Input Register, Feedback Register, and MBR. It is implemented by an array of multiplexers. *Shifters 1 and 2*, which select desired outputs from MBR. Shifter 1 is necessary since the number of the state variables can be larger than w. Shifter 2 is necessary since the number of the external outputs can be larger than w. *Control Circuit* that consists of counter and control memory generates signals for Programmable Interconnection Network and the shifters.

We can estimate the amount of hardware as follows:

Total number of data inputs: $N_{\text{total}} = N_{\text{primary inputs}} + N_{\text{FF}} + w$, and N_{FF} is the number of state variables. The number of control variables for the multiplexer is given by $\lceil \log_2 N_{\text{total}} \rceil$. Let $N_{\text{memory inputs}} = 16$; $N_{\text{memory outputs}} = w = 16$; $N_{\text{primary inputs}} = 36$; $N_{\text{FF}} = 21$; $m = 23$. Then we have $N_{\text{total}} = 36 + 21 + 16 = 73$; $\lceil \log_2 N_{\text{total}} \rceil = \lceil \log_2 73 \rceil = 7$. Thus, in Programmable Interconnection Network, we need 16 multiplexers with 73 data inputs and 7 control inputs.

In addition, we need control storage for Programmable Interconnection Network, and Shifters 1 and 2. The control words consist of 17 bits, and the number of words is equal to the maximum number of cells. Thus, the sizes of additional circuits are smaller than memory.

VI. Conclusion

In this paper, we presented the LUT ring, a new memory-based realization of sequential circuit. It requires much smaller memory than conventional realization of sequential circuits. Experimental results using MCNC benchmark demonstrate this.

Acknowledgment

This work was supported by a grant from the Japanese Ministry of MEXT via Kitakyushu innovative cluster project, the Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS), and grant of the Takeda Foundation.

References

[1] R. K. Brayton, "The future of logic synthesis and verification," in S. Hassoun and T. Sasao (eds.), *Logic Synthesis and Verification*, Kluwer Academic Publishers, 2002.

[2] S. D. Brown, R. J. Fancis, J. Rose, and Z. G. Vransic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[3] C. H. Clare, *Designing Logic Systems Using State Machines*, McGraw-Hill, New York, 1973.

[4] M. Davio, J.-P Deschamps, and A. Thayse, *Digital Systems with Algorithm Implementation*, John Wiley & Sons, New York, 1983.

[5] D. Green, *Modern Logic Design*, Addison-Wesley Publishing Company,1986.

[6] MCNC-benchmark function set: http://www.cbl.ncsu.edu/

[7] Y. Iguchi, T. Sasao, M. Matsuura, and A. Iseno, "A hardware simulation engine based on decision diagrams," *Asia and South Pacific Design Automation Conference (ASP-DAC'2000)*, Jan. 26-28, Yokohama, Japan, pp. 73-76.

[8] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[9] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *International Workshop on Logic Synthesis (IWLS-2001)*, Lake Tahoe, CA, June 12-15, 2001, pp. 225-300, also, "Realization of multiple-output functions by reconfigurable cascades," International Conference on Computer Design (ICCD-2001), pp.388-393, Sept. 2001.

[10] T. Sasao and M. Matsuura, "A method to decompose multiple-output logic functions," *Proc. of 41st Design Automation Conference*, June 2004 (Accepted for Publication).