Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 3, Number 1, pages 13–19, January 2014

Implementation of Tree Arbiter for FPGA and Metastability Analysis

Manami OGINO
Graduate School of Systems Engineering,
Okayama Prefectural University,
111 Kuboki, Soja-shi,
Okayama, Japan
ogino@circuit.cse.oka-pu.ac.jp

Tomoyuki YOKOGAWA
Yoichiro SATO
Kazutami ARIMOTO
Okayama Prefectural University,
111 Kuboki, Soja-shi,
Okayama, Japan
t-yokoga@cse.oka-pu.ac.jp
sato@cse.oka-pu.ac.jp
arimoto@cse.oka-pu.ac.jp

Masafumi KONDO
Kawasaki University of Medical Welfare
288 Matsushima, Kurashiki-shi,
Okayama, Japan
m-kondo@me.kawasaki-m.ac.jp

***Abstract*—We propose an implementation of an asynchronous arbiter for Field Programmable Gate Array (FPGA) which can reduce failures caused by metastability of RS flip-flop (RSFF). We adopt a binary tree structure of 2-input arbiters, which is called a tree arbiter. The tree arbiter has hierarchical structure of 2-input arbiters, in which each 2-input arbiter selects one of the two requests progressively. The tree arbiter can prevent increase of latency and has also an advantage in throughput. The 2-input arbiter has glitch killer circuits in order to prevent the failures caused by metastability of RSFF. However, it is impossible to implement the glitch killer circuits into FPGA. Therefore, we propose a method for preventing such failures applicable to FPGA. We also analyze the metastability of RSFF implemented with a crossed connection of logic blocks. Finally we generate a circuit model from implemented FPGA in order to take place an analysis by SPICE.**

I. Introduction

In case that resources, such as a bus or a memory, are shared by independent processing elements, conflicts about accesses to a resource may occur. An asynchronous arbiter (hereinafter referred to as an *arbiter*) establishes the order of access to a shared resource among asynchronous requests from processing elements[1]. Many methods have been proposed for constructing arbiters. Most of them multiply inputs of the arbiter by combining 2-input arbiters[2].

2-input arbiters have two stable states corresponding to the choices of the inputs. If two requests arrive at an arbiter within a short time, the output may oscillate or keep medium voltage for uncertain time before the arbiter reaches the stable states. These intermediate states are called *metastable* states and may cause erroneous behaviors. To avoid the errors, a cascading connection of glitch killer circuits inhibiting propagation of the metastability is established at the output of RS flip-flop (RSFF) in the 2-input arbiter[3].

Field Programmable Gate Array (FPGA) has been developed to implement a synchronous circuit. Recently, a number of methods have been proposed for implementing asynchronous circuits in FPGAs[4][5][6][7]. Since it is impossible to implement glitch killer circuits in FPGA, some other approaches are required to avoid the errors caused by metastability. However, there is no method for implementing the prevention of metastability in FPGA. In addition, a RSFF is implemented with a cross coupled *logic blocks* and thus metastability may occur in the RSFF. There are many studies for metastability analysis of Application Specific IC (ASIC)[3][8][9][10]. However, metastability analysis of FPGA does not exist, except for the analysis of D flip-flop (DFF) in a logic block[11][12][13].

In this paper, we propose an implementation of an arbiter composed of 2-input arbiters preventing errors caused by metastability of RSFFs. We adopt a binary tree structure of arbiters (called *tree arbiter*) which can prevent increase of latency and has also an advantage in throughput[14]. The tree arbiter has a hierarchical structure of 2-input arbiters and each 2-input arbiter selects one of the two requests progressively in each stage. We also analyze the metastability of RSFF implemented with a crossed coupled logic blocks. We derive a circuit model from implemented FPGA and analyze the model by SPICE.

In Section II, we present the hierarchical structure and operation of tree arbiter. In Section III, we present the implementation of tree arbiter for FPGA preventing errors caused by metastability. Section IV shows results of metastability analysis of RSFF implemented for FPGA.

II. Asynchronous Tree Arbiter

A. 2-input Arbiter

A 2-input arbiter, which is a basic component of a tree arbiter, is constructed by using a RSFF. The 2-input arbiter selects one request based on critical race in RSFF. However, in the case that two requests arrive within a short time, a RSFF of the arbiter could fall into a metastable state and its output keeps medium voltage. Since such behaviors may cause errors in the arbiter, a number of methods are developed to decrease the metastability or to suppress propagation of the metastability.

A *mutual exclusion element* (ME) is one of the structures for 2-input arbiters which can inhibit propagation of medium voltage. The structure of ME is shown in Fig 1.

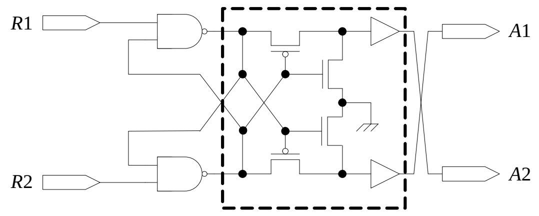


Fig. 1. The structure of a mutual exclusion element.

*R*1, *R*2 are request signals and *A*1, *A*2 are their acknowledgement signals. The dashed area which consist of four MOS transistors is called a *glitch killer* circuit. The glitch killer prevents propagation of medium voltage and is widely used to prevent errors caused by the metastability.

If *R*1 and *R*2 are asserted in a long enough interval, then an acknowledgement signal corresponding to the previous request (if the request is *R*1(*R*2) then the acknowledgement is *A*1(*A*2)) is asserted. On the other hands, if the two requests are asserted within a short time, the RSFF in the ME reaches a metastable state and outputs of its NAND gates keep medium voltage. While the RSFF remain in a metastable state, two p-MOSFETs and two n-MOSFETs turn OFF and ON, respectively. Then *A*1 and *A*2 are both disasserted and the acknowledge operation is postponed. When the RSFF leaves a metastable state, their outputs become stable and the acknowledge operation is carried out. Thus the latency increases by increasing of the propagation delay caused by the metastability.

B. Multi-input Tree Arbiter

A tree arbiter is a competitive arbiter which has a binary tree structure of 2-input arbiters. Fig 2 shows the structure of a tree arbiter. *REQi* and *ACKi* ($1 \leq i \leq N$) represent request and acknowledgement signals, respectively. A *tree module* (TM) is obtained by adding to the ME a controller operating a request to upper stage and an acknowledgement from upper stage in the tree structure. Fig 3 shows the structure of a tree module.

The AND gate labeled *C* is the Muller C-element, which is a commonly used asynchronous logic component. The output of the C-element reflects its inputs if the two inputs have a same logical value. Otherwise the output remains the value in the previous state. *OREQ* and *OACK* represent request to upper TM and acknowledgement from upper TM, respectively.

First, inputted request signals *REQi* ($1 \leq i \leq N$) are selected by TMs in bottom stage. The arbitration is performed by ME of each TM. After the arbitration is completed, the TM sends a request to upper TM by asserting *OREQ*. If the RSFF in the ME falls into a metastable state, the ME waits for metastability to resolve. The TM in next stage performs the arbitration process in the same way. Thus the arbitration processes are performed by the TMs in $\log_2 N - 1$ stages hierarchically. The ME in top stage selects one of the requests from lower TMs and asserts an acknowledgement signal corresponding to the request. If the input signal *OACK* is asserted, assuming *IREQ*1 is selected, output of the topside C-element is asserted and *IACK*1 is also asserted. Then *OACK* of the corresponding TM in lower stage is asserted. Thus the acknowledgement processes are performed by the TMs in $\log_2 N - 1$ stages hierarchically. When the TM in bottom stage completes the acknowledgement, *ACKj* is asserted and the request of the corresponding device is acknowledged. Note that C-elements of TMs in each stage keep the logic values representing selected requests.

After resources are occupied by the selected device, request signal *REQj* is disasserted and *OREQ* of the TM in bottom stage is also disasserted. If another request signal has been asserted, then the ME selects this request immediately. Since the output of the ME is gated by C-elements, *OREQ* is ensured to be disasserted. *OREQ*s of TMs in the succeeding stages are disasserted in the same way, and then the output of ME in top stage is disasserted. By disasserting the output of the ME, *OACK*s of TMs in downward stages are disasserted and finally *ACKj* is disasserted.

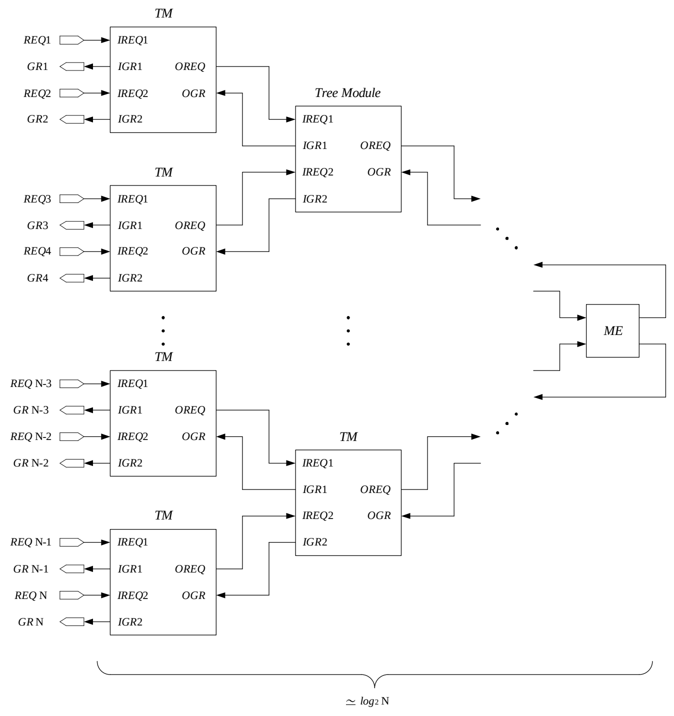


Fig. 2. The structure of a tree arbiter.

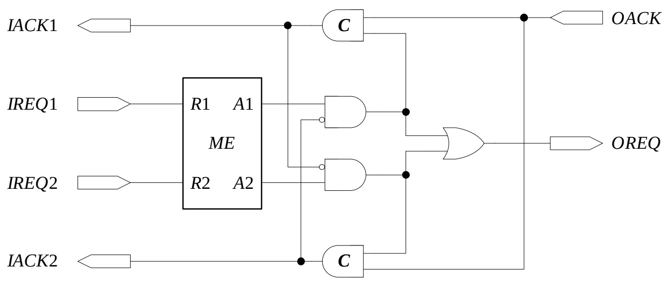


Fig. 3. The structure of a tree module.

III. Implementation of Asynchronous Tree Arbiter for FPGA

A. Improvement of tree module considering metastability

Since the tree arbiter carries out the arbitration in each stage hierarchically, metastability which occurs in each stage cannot be avoided. As mensioned in Section II, the effect of metastability is limited to increase of the propagation delay on TM and erroneous behavior does not happen. However, the glitch killer circuit shown in Fig. 1 cannot be implemented on FPGA. Thus the output of ME may remain medium voltage and it propagates to *OREQ* immediately. Such metastability may cause various erroneous behaviors. For example, if the metastable state remains until *OACK* is asserted, then both *IACK*1 and *IACK*2 can be asserted and two requests are acknowledged simultaneously. In such case, both C-elements store 1 and these requests are gated by the AND-gates, and then *OREQ* is disasserted. This will disassert *OACK* and interrupt the occupancy of resource. In this chapter, we show a method to decrease this type of errors caused by metastability.

As is clear from the logic of *OREQ* in Fig. 3, the presence of conflict on TM in each stage depends on only the presence of requests from the former stage, and not on the result of acknowledgement. Thus *OREQ* can be asserted at the time that some requests arrive at the TM. Since *OREQ* is not generated from the outputs of ME directly, ME has a low probability of propagating the metastability. In addition, even if TM falls into a metastable state, subsequent TMs can continue the arbitration process before the TM reaches a stable state and the delay caused by the metastability can be masked.

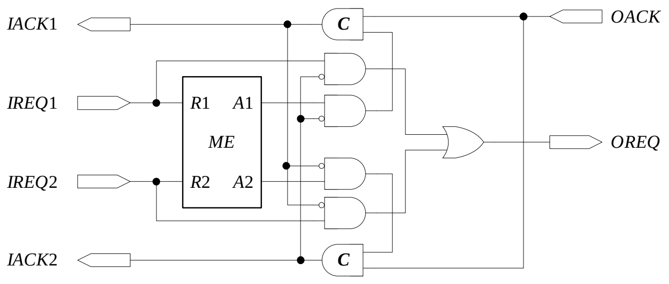


Fig. 4. The structure of a tree module with early generation of *OREQ*.

Fig. 4 shows the structure of TM where *OREQ* can be early generated. We refer to the TMs in Fig. 3 and Fig. 4 as TM$_{org}$ and TM$_{new}$, respectively. We also refer to the tree arbiters based on TM$_{org}$ and TM$_{new}$ as TArb$_{org}$ and TArb$_{new}$. Note that TArb$_{new}$ has the same structure as TArb$_{org}$ in the relation of TMs shown in Fig. 2 and TArb$_{new}$ can be obtained by replacing all the TMs in Fig. 2 with the TM$_{new}$.

In TM$_{new}$, requests *IREQ*1 and *IREQ*2 are inputted into ME and are also used to generate *OREQ*. Thus even if the metastability is caused in ME, the tree arbiter continues its arbitration process. When *OACK* is asserted, one of inputs of both C-elements is also asserted. If the metastability in ME has finished, another input of one C-element is asserted and corresponding acknowledgement *IACK*1 or *IACK*2 is asserted. Otherwise, faults may happen. However, faults caused by propagation of the metastability can be avoided.

The tree arbiter with TM$_{new}$ can continue the arbitration process if former TMs falls into a metastable state. Therefore even if a series of TMs falls into metastable state, the delay caused by the metastability is masked by the delay occurs in the subsequent TM. For this reason, the latency of TArb$_{new}$ is affected by only the delay occurs in the last ME. Thus the peak latencies L_{org} and L_{new} of n-input tree arbiter TArb$_{org}$ and TArb$_{new}$ are obtained as follows:

$$L_{org} = (t_{me} + d_{Req} + d_{Ack}) \cdot (\log(n) - 1) + t_{me}. \quad (1)$$
$$L_{new} = (d_{Req} + d_{Ack}) \cdot (\log(n) - 1) + t_{me}. \quad (2)$$

Here, t_{me} is duration of the metastability, d_{Req} is sum of delays on AND and OR gates in Fig. 3, and d_{Ack} is delay on the C-element.

B. Result of Implementation for FPGA

We designed 4-input tree arbiter for behavioral level using Xilinx ISE 9.2. In this paper we adopt Xilinx Spartan-3 as a target device, which is SRAM-based FPGA with an island structure shown in Fig. 5.

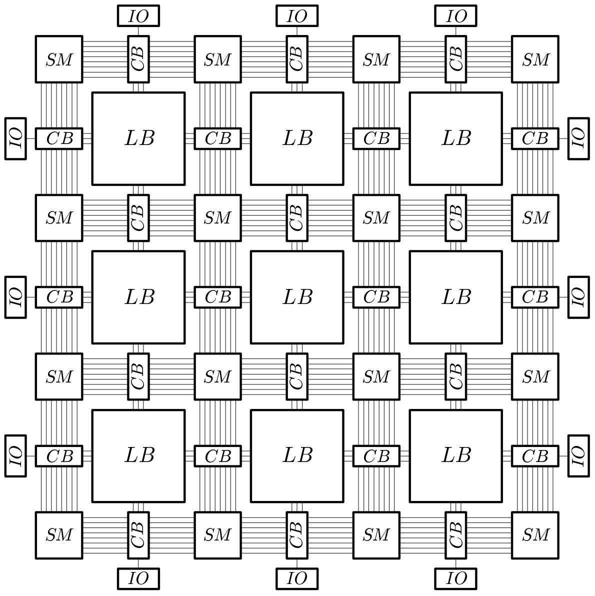


Fig. 5. SRAM-based FPGA with an island structure.

A logic block (LB) is a programmable circuit which implements a combinational circuit as a logical element and an input-output block (IO) is an interface between LB and I/O-pin of the FPGA. LB and LB or LB and IO is connected by wires, a connection block (CB) and a switch matrix (SM). CB connects LB or IO with wires programmably and SM connects vertical wires and horizontal wires programmably. LB has an original architecture for each FPGA vender. In Spartan-3 FPGA, LB consists of four slices and these slices can be programmed individually. Fig. 6 shows an outline of a structure of a slice in Spartan-3.

A slice has two lines G and F. Each line has one 4-input look up table (LUT) and one D flip-flop (DFF). G_i, F_i $(i = 1 \ldots 4)$ are inputs and X, Y are outputs of the slice, respectively. Boxes in the LUT represent SRAM cells. These

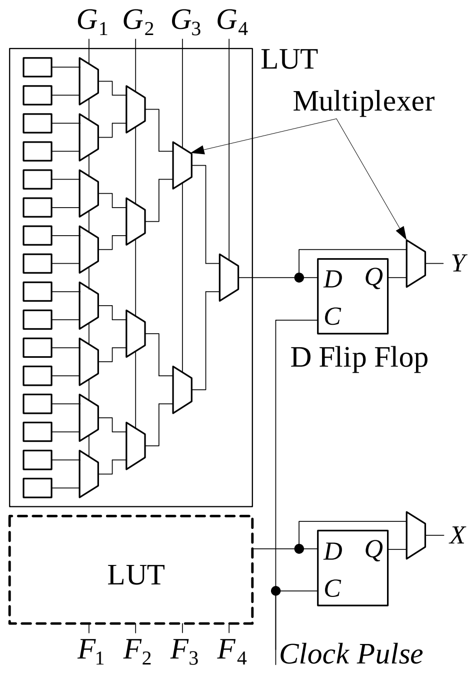


Fig. 6. Outline of a structure of a slice.

cells are set or reset according to a truth table to implement a combinational circuit. On the other hand, although a sequential circuit can be implemented using DFF, in this implementation we use RSFF by cross coupled LUTs instead of the DFF.

We provided a schematic design to implement the tree arbiter. The logical structures of ME and TM are the same as shown in Fig. 1 and 4. However, as stated above, the glitch killer circuit of ME cannot be implemented on FPGA. Thus we implemented the ME as a 2-input RSFF by omitting the glitch killer circuits. Fig. 7 shows the implementation of the 4-input tree arbiter on FPGA.

The MEs in TMs are implemented on G of $Slice1$, F of $Slice2$, F of $Slice6$ and F of $Slice7$ similar to the structure shown in Fig. 1. On the other hand, the ME in the top stage is implemented on G of $Slice3$, G of $Slice4$, F of $Slice1$ and G of $Slice6$. Although this implementation is not the same as Fig. 1, the logical functions are same. The C-elements are implemented F of $Slice4$, F of $Slice3$ and G,F of $Slice5$.

To check the operation of the implemented 4-input tree arbiter, we assigned test patterns which assert fou requests exhaustively and observed waveforms in the case that the metastability occurs. Fig. 8 shows the waveforms of request and acknowledgement signals.

The result shows that the arbitrations are carried out exclusively in some of test patterns with the metastability. Thus the improvement of a tree arbiter stated in Section III-A can prevent the effect of the metastability. However, for some cases (indicated by white arrows) multiple acknowledgements are asserted simultaneously. This seems to be because long duration of the metastability keeps the TM in a metastable state until *OACK* is asserted and two or more requests are acknowledged simultaneously.

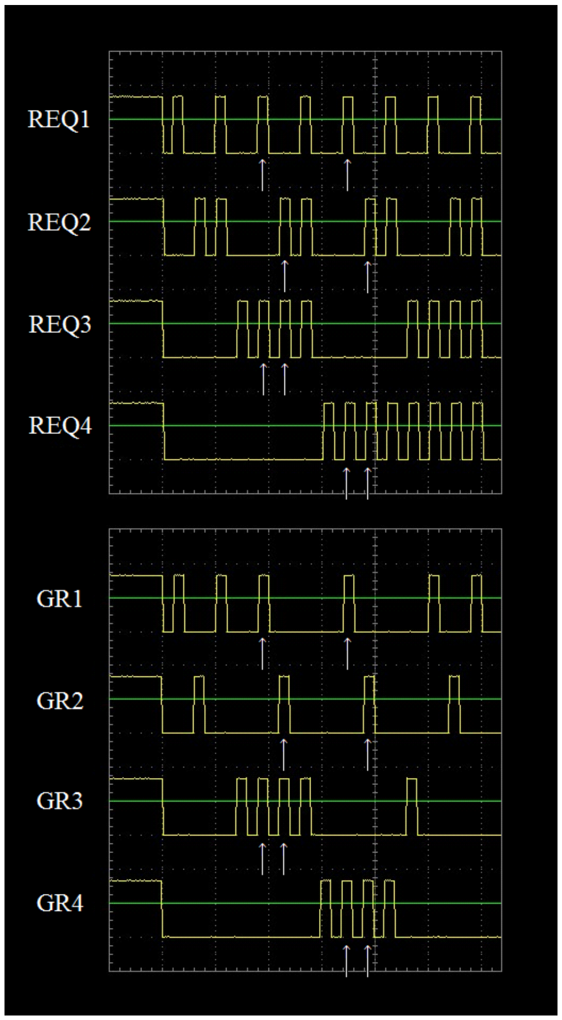


Fig. 8. Waveforms of input and output signals of the 4-input tree arbiter.

IV. Metastability Analysis of RSFF implemented in FPGA

As stated above, the implemented tree arbiter has erroneous behaviors. Such faults are caused by the behavior of RSFF in a metastable state. Since the signals inside FPGA cannot be observed, we use SPICE to analyze the metastability of RSFF implemented on FPGA.

A. Model of RSFF implemented by cross tie of logic blocks

To carry out circuit analysis, SPICE requires a circuit model of RSFF. The structure of slice is as shown in Fig. 6. We adopt structure based on path MOSFETs for a model of a multiplexer. Fig. 9 shows the structure of SM. The MOSFET matrix described in Fig. 9 is located the cross point of vertical and horizontal wires. Blocks connected to MOSFET gates indicate SRAM cells. Fig. 10 shows the structure of CB. The higher part of the CB is used for inputs and one of the wires is selected by a multiplexer. The lower part of the CB is used for outputs and connect to one of the wires by path MOSFET.

In the case that RSFF is used to process the arbitration, the inputs of the RSFF is asserted almost simultaneously. LUT

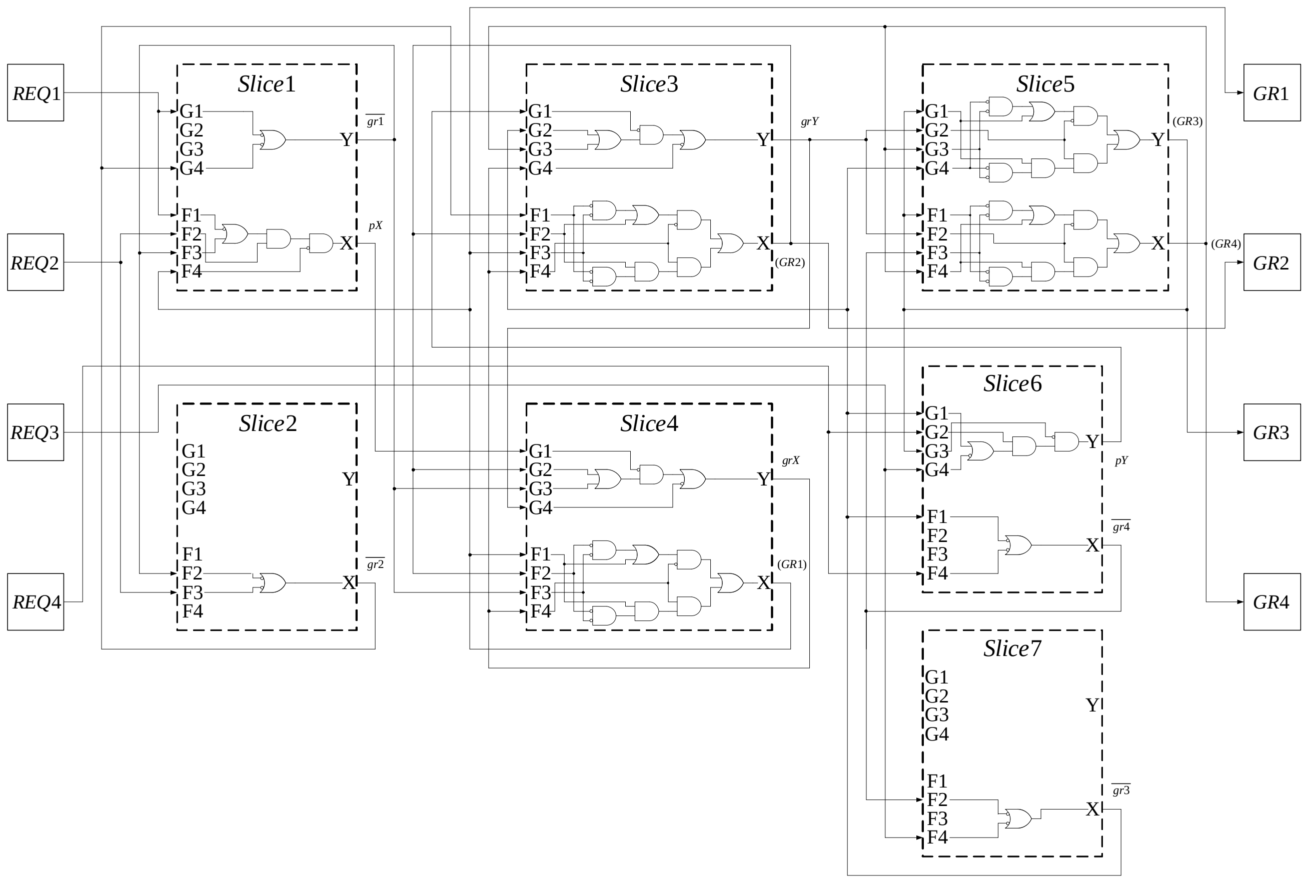


Fig. 7. Implementation of the 4-input tree arbiter on FPGA.

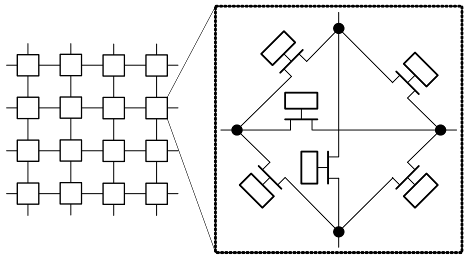


Fig. 9. The structure of switch matrix.

then selects cell SRAM which is set if the feedback input is 0 and otherwize selects SRAM which is reset and outputs through the pass MOSFET. Thus LUT is equivalent to 3-state buffer which is always turned on and modeled shown in Fig. 11.

The path MOSFET is a MOSFET which constructs a multiplexer and the number of MOSFETs depends on the implementation. SM and CB can be modeled as a cascading connection of path MOSFETs.

Thus RSFF can be modeled as a circuit shown in Fig. 12. Here, the buffers correspond to those in CB, and cascading connections of path MOSFETs correspond to the MOSFETs in SM and CB.

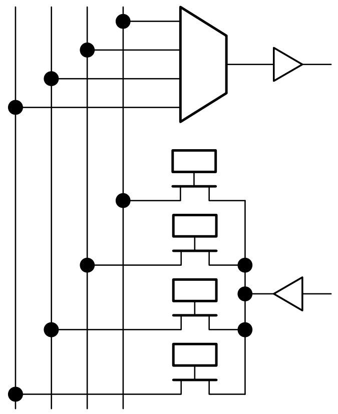
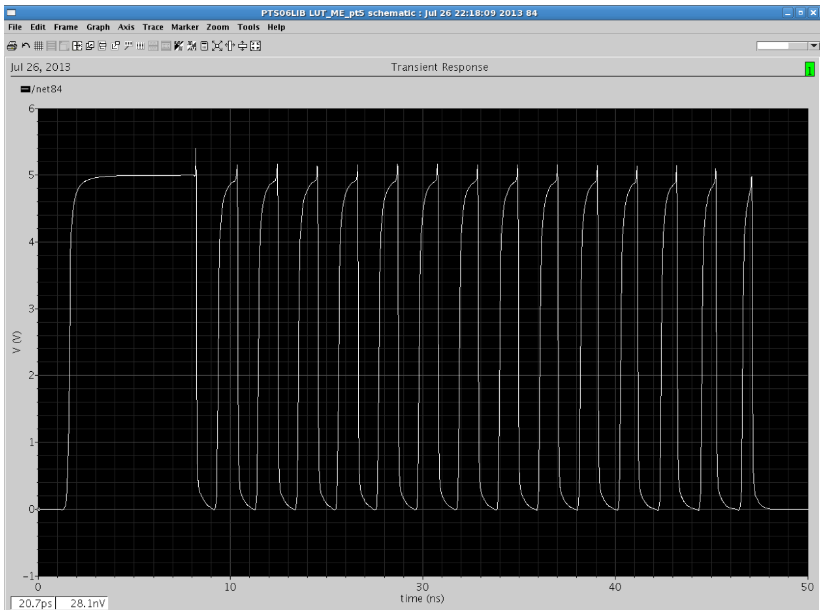


Fig. 10. The structure of connection block.

B. Result of Analysis by SPICE

We used $0.6\mu m$ rule models bsim3v3n and bsim3v3p as SPICE parameters.

We change the difference of time (referred to as Δt) when two signals inputted to RSFF change and simulated the outputs of the RSFF. We show an example of output waveform in Fig. 13, where the number of path MOSFETs is 8 and Δt is 0.01nsec. Here, the horizontal axis indicates time and the


PTS06LIB LUT_ME_pt5 schematic : Jul 26 22:18:09 2013 84
File Edit Frame Graph Axis Trace Marker Zoom Tools Help
Jul 26, 2013
Transient Response
/net84
V (V)
time (ns)
20.7ps
28.1nV

time difference between two inputs ⟶

Fig. 13. An example of waveform with the metastability.

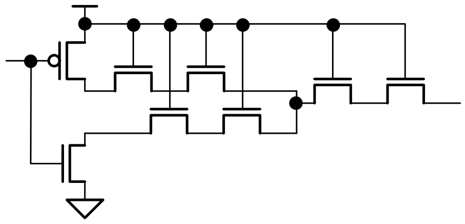


Fig. 11. The LUT model used for simulation.

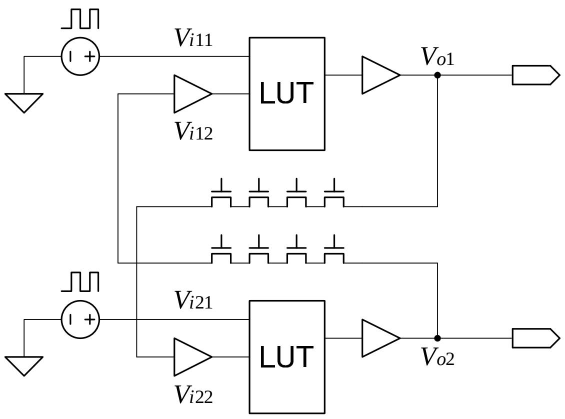


Fig. 12. The RSFF model used for simulation.

vertical axis indicates output voltage.

As shown in the Figure, this waveform oscillates. Oscillation may occur in the condition that the gain of the feedback loop is greater than or equal to 1 and the phase is an odd multiple of π. That RSFF obtains sufficient loop gain from 10 inverters on the feedback loop and its path MOSFETs satisfies the condition of the phase. Thus cross coupled LUTs may increase the probability of such oscillation. It also enlarges the difference of outputs of RSFF and then, even if the glitch killer circuit can be implemented, makes it hard to prevent the propagation of the metastability. Constrast to that, the proposed tree arbiter never causes such propagation to upper TM and has an advantage on decreasing faults.

Fig. 14 and 15 show the propagation delays for various values of Δt. The horizontal and vertical axes indicate Δt

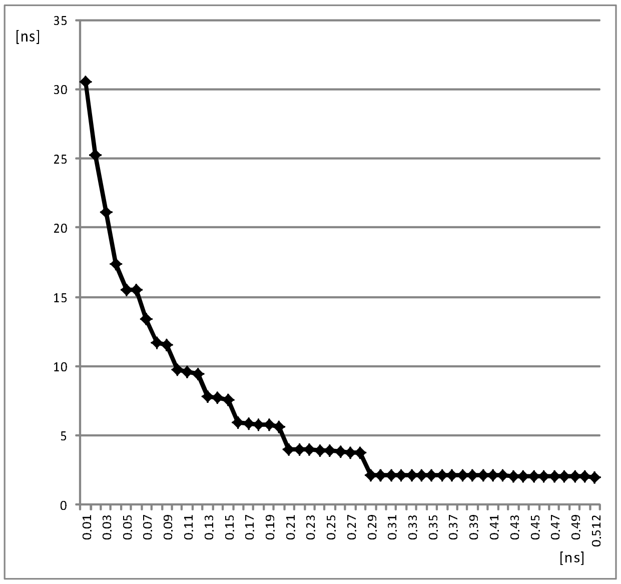


Fig. 14. Relation of Δt and propagation delay(# of path MOSFETs is 4)

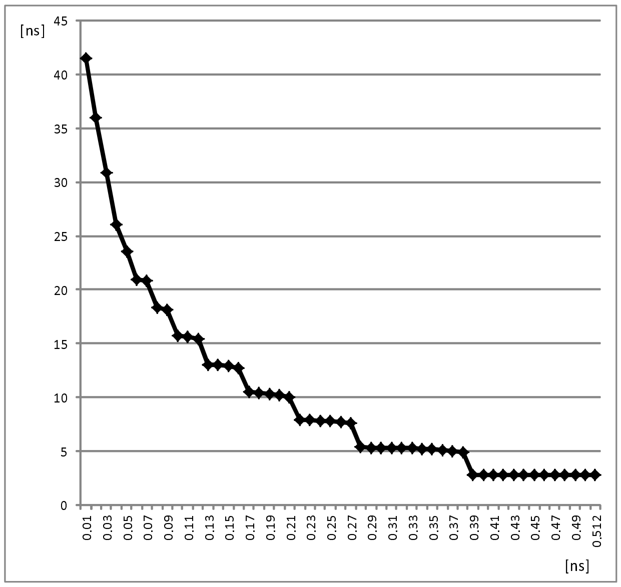


Fig. 15. Relation of Δt and propagation delay(# of path MOSFETs is 8)

and the propagation delay, respectively. The propagation delay represents the time from the inputs change until the output voltage becomes stable. In the case that output is stable to 0 or 1, we consider the output voltage is stable at the time when output becomes 10% or 90% of the power voltage last time, respectively. As Δt increases, the propagation delay becomes smaller discretely. This is because the waveform is oscillating then.

With increasing the number of path MOSFETs, the propagation delay time increases in case that metastability occurs (for example, when $\Delta t = 0.01$[sec], the propagation delay time increases about 10[nsec]). If Δt has the same value, the number of oscillations also stays unchaned and the oscillation period is increased due to the increase of propagation delay time caused by the path MOSFETs. This seems to be because that the path MOSFET has no gain factor and has only inertial delay factor. Since the path MOSFET increases the propagation delay time, the number of path MOSFETs should be smaller as possible.

V. Conclusion

In this paper, we propesed an implementation of an asynchronous tree arbiter on FPGA. This implementation can reduce faults caused by metastability without glitch killer circuits. In our method, TMs in each stage perform arbitration and send requests to the upper TMs at the same time. We also analyze the metastability of RSFF which is implemented by a cross connection of LUTs. As a result, it becomes clear that such RSFF has an oscillating output. This type of behavior cannot be prevented by a glitch killer but out method can reduce the effects of metastability.

In our study, several significant challenges still remain. We show the results of metastability analysis for tree arbiters with 4 and 8 path MOSFETs. First, it is necessary to clarify the relation between the number of path MOSFETs and performance in stable-state. We also carried out simulation and observed the behavior of RSFF. Second, to clarify the mechanism of the fault occurrence, the simulation of whole the tree arbiter must take place. Finally, evaluation of frequency of the fault occurrence is also required. These challenges need to be addressed to show efficiency of the proposed implementation for reducing faulty behaviors.

References

[1] D. J. Kinniment, et al. ,"Synchronization and Arbitration in Digital Systems", John Wiley & Sons, Ltd, 2007.

[2] K. S. Low et al. ,"Token ring arbiters: An exercise in asynchronous logic design with petri nets", Technical Report No.537, University of Newcastle upon Tyne, 1995.

[3] C. H. van Berkel et al. ,"Beware the Tree-Way Arbiter", IEEE Transactions on Solid-State Circuits, vol.34, no.6, pp.840-848, 1999.

[4] E. Brunvard, "Using FPGAs to Implement Self-Timed Systems", In Journal of VLSI Signal Processing, Vol. 6, No. 2, pp.173-190, 1993.

[5] W. C. G. Martin and A.J. Thomas, P., "An architecture for asynchronous FPGAs", In Field-Programmable Technology (FPT), Proc. IEEE International Conference, pp.170-177, 2003.

[6] L. Fesquest, R. Renaudin, "A Programmable Logic Architecture for Prototyping Clockless Circuit", TIMA Lab. Research Report, 2005.

[7] C. P.-Quoc, A.-V. D.-Duc, "New approaches to design asynchronous circuits on FPGAs", Proc. ATC 2009, pp.63-67, 2009.

[8] D. J. Kinniment et al. ,"Synchronization Circuit Performance", IEEE Transactions on Solid-State Circuits, vol.37, no.2, pp.202-209, 2002.

[9] Y. Semiat and R. Ginosar,"Timing Measurements of Synchronization Circuits", Proc. IEEE ASYNC 2003, pp.68-77, 2003.

[10] I. W. Jones and M. Greenstreet,"Synchronizer Behavior and Analysis", Proc. IEEE ASYNC 2009, pp.108-117, 2009.

[11] J. Kalisz and Z. Jachna,"Metastability tests of flip-flops in programmable digital circuits", Microelectronics Journal, vol.37, no.2, pp.174-180, 2006.

[12] Rogina, B.M. et al. ,"Metastability testing at FPGA circuit design using propagation time characterization", Proc. Design & Test Symposium (EWDTS), pp.80-85, 2010.

[13] T. Polzer and A. Steininger, "An Approach for Efficient Metastability Characterization of FPGAs through the Designer", Proc. ASYNC 2013, pp.174-182, 2013.

[14] M. Imai, et al. ,"N-way ring and square arbiters" ・訓 roc. ICCD'09, pp.125-130, 2009.