Clocking Scheme That Realizes Ballistic Signal Flow

Ushio Jimbo
Dept. of Informatics
School of Multidisciplinary Sciences, SOKENDAI
Tokyo, Japan
ushio@nii.ac.jp

Ryota Shioya
Dept. of Creative Informatics
Graduate School of Information Science and Technology, The University of Tokyo
Tokyo, Japan
shioya@nuee.nagoya-u.ac.jp

Masahiro Goshima
Systems Architecture Research Division
National Institute of Informatics
Tokyo, Japan
goshima@nii.ac.jp

Random Variation and Worst-Case Design: To date, the *worst-case design* has been generally accepted. In the worst-case design, LSIs are designed so that they work even if all the elements had the worst case delays. However, this estimation is pessimistic. The probability that the all the elements in a circuit path have the worst-case delays is extremely low. The worst-case design worked only because the worst-case delays were not so far from the typical-case ones, and will be too pessimistic with the increased gap between the typical- and the worst-case delays. We cannot expect the same performance improvements as seen in the past with the worst-case design.

To deal with this problem, a wide variety of methods have been considered. For example, *statistical static timing analysis (SSTA)* is introduced to relax the too pessimistic estimation. *Timing fault detection* is a circuit-level technique to avoid the problem.

Timing Fault Detection: A *timing fault* is a transient fault caused by dynamic variation in the circuit delay. In the worst-case design, LSIs are designed so that a timing fault does not occur even in the worst cases. Therefore, timing faults can only occur in unexpected situations such as thermal runaway caused by a failure of the thermal sensor.

The timing fault detection techniques can avoid the problem of worst-case design, especially with *dynamic voltage and frequency scaling (DVFS)*, in the following way. A device operates with a lower voltage and/or a higher frequency than given by the worst-case design. As a matter of course, a timing fault occurs, but it is detected by a detection circuit, such as a *Razor FF* [1]–[3]. Then, the device is recovered to a fault-free state. The voltage and the frequency should be configured so that the fault rate becomes low enough. The device operate with the actual delay according to the actual operating environment of the individual device, and to the actual performance of the individual elements of the device. In this way, timing fault detection can transcend the worst-case design.

Razor can mitigate the varation of actual delay *in a stage*. However, it do not care about that *inter stages*. That is a room for improvement.

Clocking Scheme That Realizes Ballistic Signal Flow: We proposed a novel clocking scheme enabling *dynamic time borrowing*, which realizes ballistic signal flow. Figure 1 shows a system diagram for our scheme. The architecture of our scheme is as follows:

1) This scheme is based on two-phase latch scheme.
2) Pipeline latches with longer path delays are replaced by Razor-like latches to detect timing faults.
3) The recovery from a timing fault is based on the *architecture state* protected from timing faults in the same way as Razor [2].
4) The error network is added to gather the e output of each Razor latches along with the pipeline stages to disable the update of the architecture state with faulty results.
5) An empty stabilize stage is inserted.

Ballistic Signal Flow: Our clocking scheme can enable time borrowing of the actual delay between neighboring stages, which means no faults are reported until the *accumulation of delay* violates the timing constraint. The circuit with our scheme behaves as a long combinational logic that spans multiple stages. According to the law of large numbers, the long combinational logic averages the process variation of the element delay in the logic. This may reduce the probability of timing-faults compared to that of Razor. Comparing with a conventional scheme using single-phase flip-flops, our clocking scheme can double the minimum cycle time relative to the critical path delay of the circuit.

Method to Avoid Short Path Problem: Razor has short path constraint. In order to solve the constraint, Razor by inserting delays into short path. On the other hand, in our scheme, to enable dynamic time borrowing between pipeline stages, we apply an method as follows: In Figure 1, short and critical paths the short path and the critical path of the logic

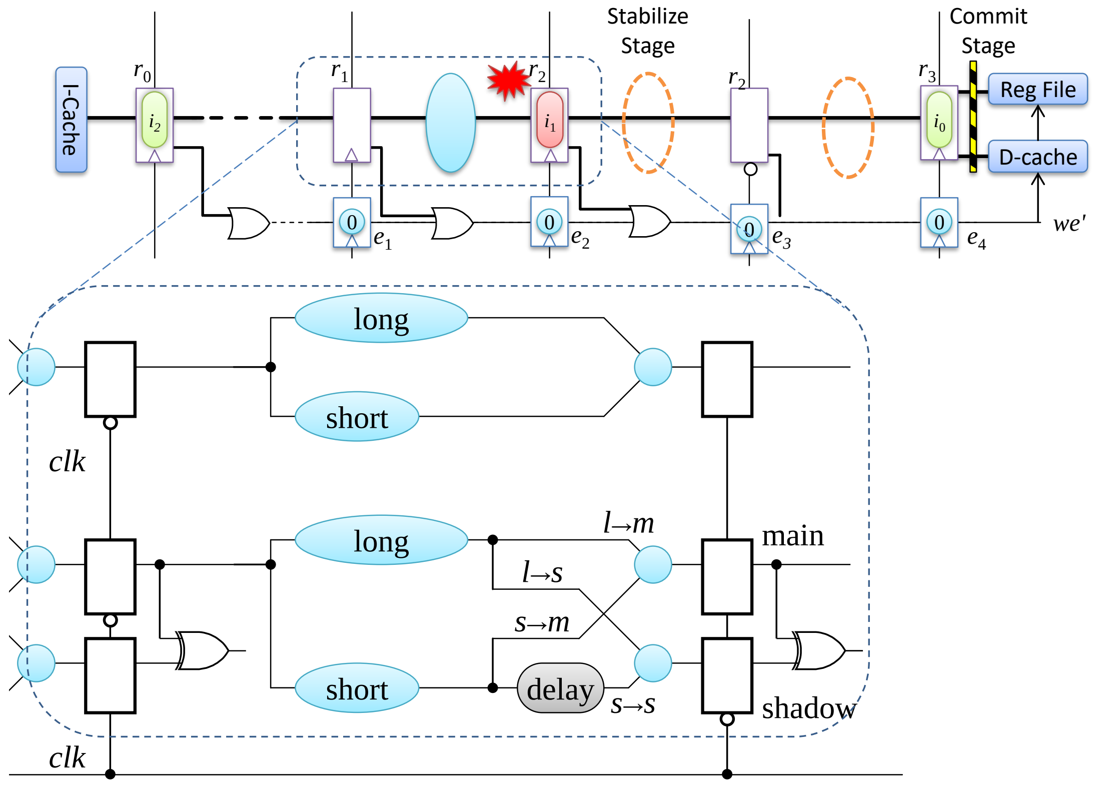

Fig. 1: Clocking Scheme That Realizes Ballistic Signal Flow.

merge at the gate (indicated by the circle) in the figure, and then are connected to the latch. In this case, duplicate gates ○ to join, connect each to main and shadow. Then insert the delay only in the short path leading to the shadow. By doing so, you can achieve both of the following:

1) The short path problem peculiar to Razor can be solved by inserting a delay in the short path leading to the shadow.
2) Because delays are not inserted in the path leading to the main, it is avoided that the effective delay in the case where the short path is activated is elongated. This will maximize the effect of the dynamic time borrowing.

Application of Our Scheme to Rocket Processor: We apply this clocking scheme to Rocket [4] and implement it in an FPGA to evaluate the scheme. We evaluate the effect of this scheme by measuring its rate of timing faults by frequency and the maximum frequency where the processor can operate.

For the application, we need to convert Rocket as follows: (1)Rocket executes instruction in out-of-order after some instructions. However, if a preceding instruction overtaken causes a timing fault, the circuit cannot recover correctly. To avoid this, we need to disable out-of-order commit of Rocket. (2)We need to convert it from FF to two-latch scheme. We use our algorithm utilizing maxflow.(3)We need to replace the end FF of timing risky path with Razor-like latch and add the circuit with architectual recovery. We show the conversion is mostly automatable and increases design complexity very little.

References

[1] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, “Razor: A low-power pipeline based on circuit-level timing speculation,” Dec. 2003, pp. 7–18.
[2] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, “RazorII: In situ error detection and correction for PVT and SER tolerance,” vol. 44, no. 1, pp. 32–48, Jan. 2009.
[3] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, “A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation,” Feb. 2010, pp. 284 –285.
[4] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, “The Rocket chip generator,” EECS Dept., UCB, Tech. Rep. UCB/EECS-2016-17, Apr. 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html