Design of VLSI Digital Filters for Tolerating Transient Timing Errors*

Hsin-Chou Chi[1,2], Hsi-Che Tseng[1] and Yih-Kai Wang[1]

[1]Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
[2]Institute of Electronics Engineering, National Dong Hwa University, Hualien, Taiwan
hcchi@mail.ndhu.edu.tw

Abstract - As the feature size of chips shrinks with semiconductor technology advancing, the size of transistors and their operating voltage keep decreasing. One of the major problems with advanced semiconductor technology is timing errors caused by process variation and noises. With such problems, conventional worst-case designs suffer poor system performance. This paper proposes an aggressive design technique for VLSI digital filters for tolerating transient timing errors. When a timing error occurs, the system reconfigures the buffer cells of digital filters with little performance degradation. We have applied the technique to two example digital filter designs, including an FIR filter and an IIR filter. The implementation results show that our proposed designs achieve tolerance of transient timing errors with reasonable cost.

Index Terms – Timing errors; error-resilient design; digital filters; VLSI design.

1. Introduction

With ever advancing of semiconductor technology, the size of transistors and their operating voltage keep decreasing. One of the major problems with advanced semiconductor technology is that integrated circuits are more and more susceptible to process variation and noises. Hence, it is more difficult to predict the timing behavior of the system due to delay variation of circuits [1].

Most current designs of integrated circuits are based on the worst-case scenario. Such conservative designs consider all possible delay variation in order to guarantee safe system operations. However, this conventional approach results in poor system performance due to serious process variation and noises with advanced semiconductor technology. Hence, aggressive methodologies have been proposed for robust system design which can tolerate timing errors [2,3,4].

In the *Razer* project, better-than-worst-case design was proposed for processor pipelines with double sampling of data signals [2]. When a timing error occurs, the whole pipeline is halted for a clock cycle. *T-error* is another research work that deals with error-resilient link design in network-on-chip architectures [3,4]. With similar double sampling technique, the design in *T-error* can tolerate timing errors for network links. Mitra et al. proposed to use C-elements for robust design of systems [5,6]. However, their approach is for tolerance of soft errors instead of timing errors.

In this paper, we propose the robust designs of VLSI digital filters with tolerance of transient timing errors due to process variation and noises. In the next section, previous work on design techniques for VLSI systems for tolerating timing errors is surveyed. In Section III, the design of our proposed VLSI digital filters for tolerating transient timing errors is presented. With our designs, the transient timing error occurring to digital filters can be tolerated efficiently. The design and implementation of two example timing-error-resilient VLSI digital filters are described in Section IV, including an FIR filter and an IIR filter. The results are compared with designs without tolerance of timing errors. A summary and conclusions are given in the final section.

2. Design for VLSI Systems with Timing Errors

The rapid progress of semiconductor manufacturing has made the design of reliable VLSI circuits more and more challenging. One of the main problems with modern and future chip design is process variation and noises. Such problems may cause timing errors for VLSI circuits. When timing errors occurs, the latches or registers in the circuits may get the instable or wrong data signals due to not sufficient delay of the clock cycle.

One of the design techniques for tolerating timing errors is to use double sampling of input data. Besides the main clock, there is a delayed clock to sample the input data, which has sufficient time to catch the correct data when there is a timing error for the main clock. Fig. 1 shows the buffer cell for the *T-error* systems, which is employed in the link design for network-on-chip architectures [5,6].

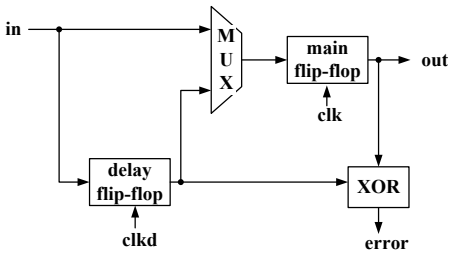

Fig. 1 Buffer cell used in T-error.

There are two flip-flops in the buffer cell in Fig. 1: main flip-flop and delay flip-flop. The main flip-flop samples the input data at the edge of the system clock *clk*, while the delay flip-flop does that at the edge of the delay clock *clkd*. The

* This work is partially supported by National Science Council, Taiwan, under Grant NSC100-2221-E-259-028.

clock edge of *clkd* arrives later than *clk* by a portion of one clock cycle, e.g. two thirds or three fourths of a cycle. When a timing error occurs, the data written into the main flip-flop is incorrect due to not sufficient cycle time. However, the delay flip-flop is able to catch the correct data. With the XOR gate to indicate the occurrence of the timing error, the cell enters the delay mode. When in the delay mode, the cell uses the output of the delay flip-flop instead of the *in* through the multiplexer as the input data for the main flip-flop. Note that there is an invalid data sent out to the output before the correct one.

We can aggregate a group of the above *T-error* cells to compose a buffer stage for the network-on-chip link transmission. For example, if the link is 16 bits wide, then a buffer stage of the link consists of 16 *T-error* cells. Fig. 2 shows the organization of the link design which is a *b*-stage pipeline and *w* bits wide. The error signals from the cells in the same stage are OR'ed together and the output signal is sent to the corresponding control circuit to indicate there is a timing error in this stage. The control circuit directs the multiplexers in the corresponding stage. The control circuit also informs the downstream stage whether the data sent is valid or not for the timing error.

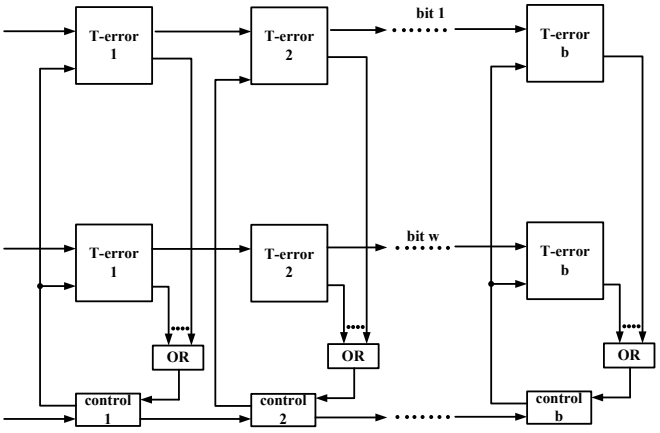

Fig. 2 T-error link with *b* pipeline stages and *w* bits wide.

The *T-error* system is aimed at solving the timing error problem with network-on-chip links. There is no combinational logic circuit between stages, and there are no irregular paths in the systems. *Razer* proposed a processor pipeline design which can tolerate timing errors by using the double sampling of data too. When a timing error occurs, the whole pipeline is halted for a clock cycle. There are combinational logic circuits between the pipeline stages. However, by simply halting the whole pipeline for a clock cycle, such solution hurts the pipeline performance. Their design also incorporates a dynamically adjusting clock rate mechanism, which can be complex and is only suitable for large-scale systems.

3. Digital Filters for Tolerating Transient Timing Errors

In this paper, we propose a design technique for VLSI digital filters which can tolerate transient timing errors. Our approach uses double sampling of data, too. However, our design allows using combinational logic circuits between adjacent pipeline stages while it avoids halting the clock or adjusts the clock rate. The digital filters design typically has forwarding or feedback datapaths in the system, and hence is more difficult to solve the timing error problem.

In our design, a pipeline buffer is modified such that it can detect a timing error. The associated control circuit can reconfigure the modified buffer and correct the data in one clock cycle. Furthermore, the control circuit will inform the downstream stage that a timing error occurs and the data transmitted is invalid. Fig. 3 shows the simple organization of the error-resilient design for a pipeline buffer, while Fig. 4 shows the detailed circuit in the modified pipeline buffer.

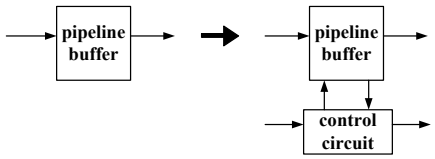

Fig. 3 Modified pipeline buffer with control circuit.

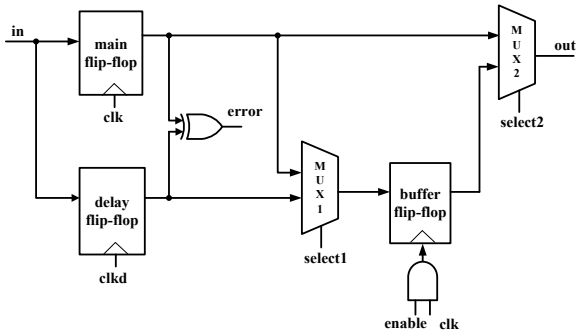

Fig. 4 One-bit pipeline buffer for tolerating timing errors.

There are three flip-flops in a one-bit pipeline buffer for tolerating timing errors, shown in Fig. 4. The main flip-flop and the delay flip-flop are driven by two different clock signals with the same frequency. The input data is sampled at the delay flip-flop with *clkd* that is delayed by a portion of a clock cycle from the main clock signal *clk*. It is supposed that the delay flip-flop is able to catch the correct input data when the timing error occurring since it has much longer time than the clock period. When a timing error occurs, the outputs of the two flip-flops do not conform to each other, and the XOR gate indicates the error. When any one-bit buffer in the same pipeline stage indicates an error, the control circuit for the stage will inform and reconfigure all buffers in the same stage. The buffer flip-flop is not operational when there is no timing error in order to save power.

When a timing error occurs to any one-bit buffer, the *enable* signal will be asserted by the control circuit for the corresponding stage, and the buffer flip-flop starts to operate. The *select1* signal controls the input multiplexer MUX1 for

the buffer flip-flop. When a timing error occurs in the same stage, the *select1* will make MUX1 to choose the output of the delay flip-flop as the input of the buffer flip-flop. The *select2* signal controls the pipeline buffer output. When a timing error occurs, the *select2* will make MUX2 to choose the output of the buffer flip-flop to be the output of this one-bit buffer. When a timing error occurs to a stage, the control circuit will also inform the downstream stage that the data transmitted is invalid.

We have designed different control circuits for the pipeline buffer in the digital filters for tolerating transient timing errors. Such control circuits include regular control circuit, forward control circuit, and feedback control circuit. Fig. 5 shows the regular linear datapath with its associated control circuit. There are two input control signals for the regular control circuit: *error* and *valid-in*. The *valid-in* indicates whether the currently transmitted input data from the upstream stage is valid or not. There are four output control signals for the regular control circuit: *select1*, *select2*, *enable*, and *valid-out*. The *valid-out* is used to inform the downstream stage whether the currently transmitted output data is valid or not.

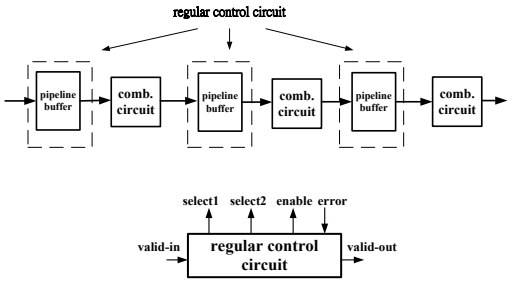

Fig. 5 Regular linear datapath and its associated control circuit.

Forward datapath is common in many digital filter designs. In a forward datapath circuit, two different datapath with the same data flow direction join in the same stage. Fig. 6 shows the forward datapath and its associated control circuit. The two pipeline buffers indicated by dash-line rectangular are the related forward stages. The forward control circuit is used with each of the two related forward pipeline stages for tolerating timing errors.

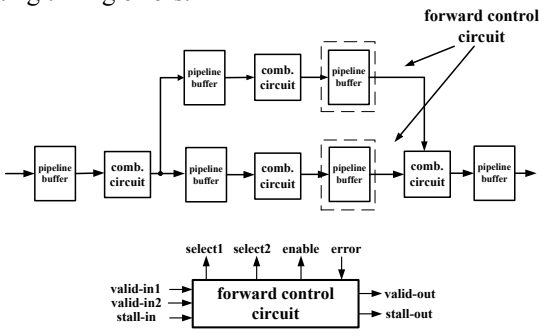

Fig. 6 Forward datapath and its associated control circuit.

There are four input control signals for the forward control circuit: *error*, *valid-in*, *valid-in2*, and *stall-in*. The *valid-in1* is the valid-out from the upstream stage. The *valid-in2* is the *valid-out* from the related forward stage. The *stall-in* is the *stall-out* from the related forward stage. This signal indicates that there is a timing error in the related forward stage. When the *stall-in* is asserted, the buffer stage needs to also switch to the delay mode in order to coordinate with the other forward stage. There are five output control signals for the forward control circuit: *select1*, *select2*, *enable*, *valid-out* and *stall-out*.

Fig. 7 shows that the *valid-out* signals of the two forward stages are OR'ed to generate the *valid-in* for the following stage. This indicates that only when both previous forward stages have valid data output, the current stage can have valid data input. In other words, if any of the two forward stages produce invalid data, the current stage will not receive valid data input. Note that an OR gate is used instead of an AND gate since "0" value indicates valid data in our design.

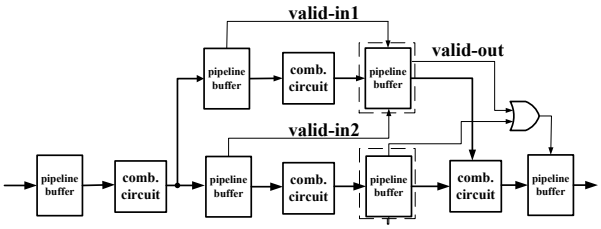

Fig. 7 Valid-out signals combining for the forward control circuits.

Feedback datapath is also common in many digital filter designs. In a feedback datapath circuit, two different datapath with the opposite data flow direction join in the same stage. Fig. 8 shows the feedback datapath with two related stages. The two pipeline buffers indicated by dash-line rectangular are the related feedback stages. The feedback control circuit is used for one of the related stages. The other related stage uses feedback joining control circuit. Fig. 9 shows the feedback control circuit and the feedback joining control circuit.

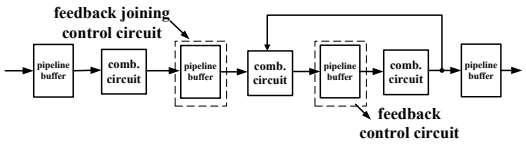

Fig. 8 Feedback datapath.

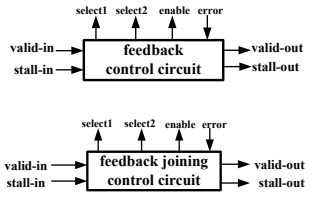

Fig. 9 Feedback control circuit and feedback joining control circuit.

A digital filter may have both forward datapath and feedback datapath combined. Fig. 10 shows a combined datapath. In such combined datapath, the combined control circuit consisting of feedback control and forward control is employed to handle such case.

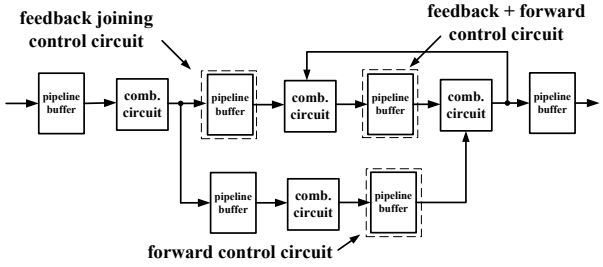

Fig. 10 Combined datapath with feedback and forward control circuit.

4. Design and implementation of example error-resilient digital filters

We have applied our design techniques in the above to example digital filters for tolerating timing errors. This section presents the results for two filters: one is an 8-tap low-pass FIR filter, and the other is a 3-parallel 2nd-order IIR filter. Fig. 11 shows the error-resilient design of the FIR filter and Fig. 12 shows the error-resilient design of the IIR filter.

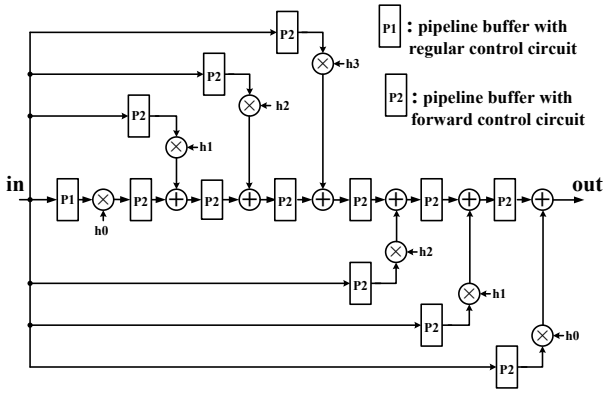

Fig. 11 Error-resilient design of the 8-tap low-pass FIR filter.

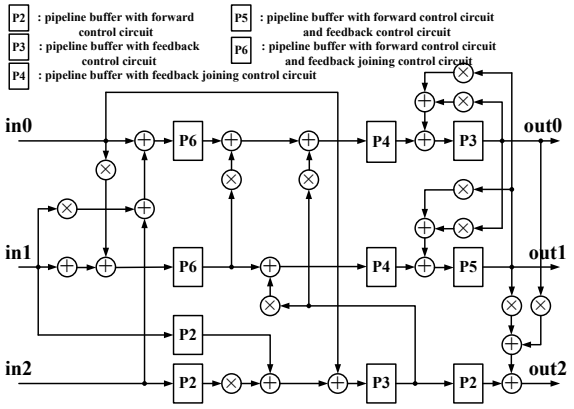

Fig.12 Error-resilient design of the 3-parallel 2nd-order IIR filter.

We have implemented the two 16-bit digital filters with both original design and error-resilient design with cell-based design flow. TSMC 0.18μm technology is used and the operating voltage is 1.62v. Table I and Table II shows our implementation results. Our results demonstrate that our error-resilient designs achieve tolerance of transient timing errors with reasonable area cost. Note that the circuit speed of the error-resilient designs is the same as the original designs. The reason for this is that the multiply and add operation in the combinational circuit dominate the clock cycle, and the delay of the extra circuit in the error-resilient design is quite small and hence is hidden.

TABLE I Implementation results for the FIR.

	16-bit FIR	16-bit error- resilient FIR
Clock rate	20.8 MHz	20.8 MHz
Area of core	293282 μm^2	551082 μm^2
Gate count	20594	38672
Power consumption	5.8 mW	7.2 mW

TABLE II Implementation results for the IIR.

	16-bit IIR	16-bit error resilient IIR
Clock rate	22.7 MHz	22.7 MHz
Area of core	523244 μm^2	776166 μm^2
Gate count	36735	54465
Power consumption	12.7 mW	14.7 mW

5. Summary and conclusions

This paper presents an aggressive design technique for VLSI digital filters for tolerating transient timing errors. When a timing error occurs, the system reconfigures the buffer cells of digital filters with little performance degradation. We have applied the technique to two example digital filter designs, including an FIR filter and an IIR filter. The implementation results show that our proposed designs achieve tolerance of transient timing errors with reasonable cost.

References

[1] R.K. Iyer, N.M Nakka, Z.T. Kalbarczyk and S. Mitra, "Recent Advances and New Avenues in Hardware-Level Reliability Support," *IEEE Micro*, vol. 25, no. 6, pp. 18-29, 2005.

[2] D. Ernst et al, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2003.

[3] R.R. Tamhankar, S. Murali and G.D. Micheli, "Performance Driven Reliable Link Design for Networks on Chips," *Proc. Design Automation Asia and South Pacific Conference*, Jan. 2005.

[4] R. Tamhankar et al., "Timing-Error-Tolerant Network-on-Chip Design Methodology," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 7, pp. 1297-1310, July 2007.

[5] S. Mitra et al., "Robust System Design with Built-In Soft Error Resilience," *IEEE Trans. Computers*, vol. 38, no. 2, pp. 43-52, Feb. 2005.

[6] S. Mitra et al., "Soft Error Resilient System Design through Error Correction," *Proc. IFIP Int. Conf. Very Large Scale Integration*, pp.332-337, Oct. 2006.