Design Methodology of Multistage Time-domain Logic Speculation Circuits

Yinan Sun Yongpan Liu Xiaohan Wang Hongliang Xu Huazhong Yang
Tsinghua National Laboratory for Information Science and Technology
EE Dept., Tsinghua University, Beijing, 100084, P.R. China
Email: {ypliu,yanghz}@tsinghua.edu.cn

***Abstract*— As variable delays are observed in the integrated circuits under different data inputs, it is expected to enhance the performance of the circuit using the average-case design methodology. This paper presents a novel approach using the time-domain multistage speculation to realize a variable-latency circuit, in which speculation points with double-sampling and check-recovery units are inserted into the critical path to enhance the performance. Furthermore, a design framework is implemented to convert a original circuit into the new one automatically. Experimental results showed that a $1.79 - 4.42$ speedup in a 64-bit ripple carry adder and up to 30.5% throughput enhancements in several ISCAS and MCNC benchmarks with reasonable area overheads.**

I. Introduction

The critical path of a combinational circuit determines the peak operating frequency. Restructuring the logic or resizing the gate may shorten the path, however at the cost of power and area. The pipeline technique can also shorten the critical path and increase the data throughput, but suffers from the longer latency. As [1] indicated, the critical paths generally take a small fraction of the entire paths and are infrequently activated, which implies that designing variable latency circuits is promising. What's more, some technologies, e.g. slack redistribution [2], [3] intended to treat the frequently-exercised and infrequently-exercised paths unequally to realize the average-case performance. [4], [5] used the telescopic unit to determine the variable clock cycles needed. As we can see, the average-case design style is receiving more and more attentions.

Among the average-case design approaches, the speculation structure adopts the check-recovery mechanism and provides more flexibility to the circuit design. There are two major kinds of speculations: functional speculation and time-domain speculation. Early works [6], [7] used the two-stage speculation structure, which has an inherent 50% limit on the performance improvement. Recently, Liu et al. [8] proposed a multistage functional logic speculation adder to overcome the limit, but it is not straightforward to extend the technique to other circuits due to the nontrivial speculation circuits design. What's more, other functional speculation techniques [9], [10] are also adopted in the limited type of specific circuits. On the contrast, the razor architecture [11] is attributed to the time-domain speculation, which is easier to be used in the general designs. However, the performance improvement of the design is limited by the detection window which has to meet the register's hold time. It should be noted that all above work needs nontrivial additional circuits, which have significant power and area overheads and a longer design cycle.

This work was supported in part by the NSFC under grant 60976032, National Science and Technology Major Project under contract 2010ZX03006-003-01, and High-Tech Research and Development (863) Program under contract 2009AA01Z130.

In this paper, we propose a multistage time-domain logic speculation method, which adds the time dimension to the multistage speculation approach. Compared with the multistage functional speculation, our method spends less areas and is more straightforward to be used in the general designs. Our contributions are listed as below:

- We proposed a multistage time-domain speculation structure and applied it in a ripple adder and some ISCAS and MCNC benchmarks. We deduced the analytical performance for the design and validated them in our experiments.
- Based on the analytical models, we implemented an automatic design framework to convert the original design into the one with the time-domain speculation. A heuristic speculation point and frequency selection algorithm is used and its effectiveness is shown by the gate-level simulation.
- Experimental results showed that the 64-bit rippler carry adder achieves a $1.79 - 4.42\times$ speedup under variable speculation stages, and up to 30.5% throughput enhancements are observed in several ISCAS and MCNC benchmarks with reasonable area overheads.

We organize the rest of the paper as follows. We presented the multistage time-domain speculation circuit in Section II. The design methodology and performance analysis of the proposed circuits are described in Section III. Section IV shows the experimental results and we concluded the paper in Section V.

II. Multistage Time-domain Logic Speculation Circuit

This section illustrates our proposed structure. We first describe the double-sampling and check-recovery unit in each speculation point and then show the architecture of the multistage speculation, in which multiple speculation points will be inserted.

A. Time-domain speculation circuit

In Fig. 1, we present a double-sampling and check-recovery architecture used in each speculation point. The Sampling Flip-flop driven by the sampling clock clk_sa samples the input and

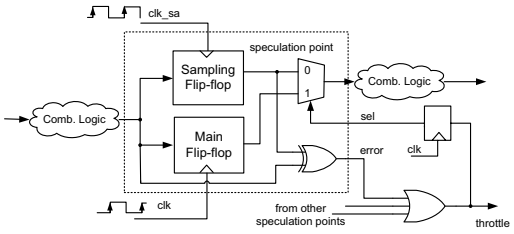

Fig. 1. Time-domain logic speculation

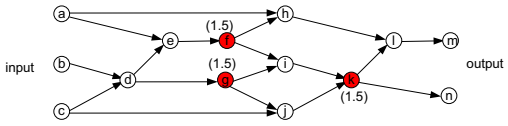

Fig. 2. DAG model of a circuit

stimulates the following logic behind the multiplexer. When the main clock clk arrives, which is later than clk_sa, the Main Flip-flop re-samples the input value, which is guaranteed to be correct by setting the clock period larger than the longest propagating delay. The signal $error$ is valid when a conflict between the re-sampled value and the previous one is detected by the XOR gate. In the next clock cycle, the selecting signal sel enables the multiplexer to transfer the re-sampled value to the following stage. In contract to the Razor approach, the contamination delay is not an issue in our design.

B. Multistage speculation

Next, we demonstrate how to construct a multistage speculation circuit from an original combinational logic. A combinational logic circuit can be expressed as a directed acyclic graph (DAG) $G(V, E)$. Each element in the set V represents a node in the circuit. The set of the edge E indicates signal paths between nodes, and the weight associated with each edge gives the logic delay between nodes. We denote the delay of a critical path between node i and j as $\tau_{i,j}$. After some speculation points are inserted into the combinational circuit, the clock frequency τ_{cyc} is determined by the followings:

$$\tau_{cyc} \geq \max\{t_i^{sa} + \tau_{i,j} + \tau_j^{spec}\} \tag{1}$$

where t_i^{sa} is the sampling time of node i ($t_i^{sa} = 0$ when node i is an input one) and τ_j^{spec} is the internal delay of the speculation point ($\tau_j^{spec} = 0$ when node j is an output one), including the setup time of the flip-flop, the delay of the multiplexer and etc.

During the normal operation, all speculation points generate a signal $error$, which is connected to a OR gate to generate a signal $throttle$. The $throttle$ signal can suspend the circuit and provide a signal sel to the multiplexer, which selects the correct value and re-computes the results. The circuit may hang on for several cycles until the $throttle$ signal is invalid. Generally, the circuit is expected to finish an operation in one cycle with careful selection of speculation points and sampling time. The performance of the circuit τ_{ave} can be expressed as the followings:

$$\tau_{ave} = \sum_{\mu} P_\mu \cdot \mu \cdot \tau_{cyc} \tag{2}$$

where P_μ is the probability of the circuit to complete the calculation in μ cycles and τ_{cyc} is the main clock cycle determined by Equation 1.

In Fig. 2, assume that all the edges have the same unit delay 1. Thus, the critical path is 7 (c-d-e-f-i-k-l-m) before the speculation points are inserted. We can select nodes $\{f, g, k\}$ as the speculation points (SP) to cut the critical path and the sampling time is given as 1.5. After inserting the SPs into the circuit, the critical path changes to $1.5 + 3 + 1 = 5.5$. By this way, the performance is enhanced.

C. Performance Estimation and Sampling Time Optimization

This subsection provides a theoretical analysis on the performance of the speculation circuit and points out how to optimize the sampling time given a SP configuration. Assuming that α_i denotes the probability of an error occurrence at the ith SP, we define the probability to observe an error happening in a speculation circuit with n SPs as P. When α in all the SPs are completely correlated, we have $P_l = \max_{i=1}^{n}(\alpha_i)$, which is the lower bound of P. When α in all the SPs are completely exclusive, we have $P_u = \sum_i^n \alpha_i$, which is the upper bound of P. [8] had proven that all the speculation error occurs independently in a ripple carry adder under random input vectors. Moreover, simulation results in the MCNC and ISCAS benchmarks showed that the independent assumption on the α is a good approximation. Under this assumption, the probability P_d to observe an error in the first cycle of the speculation circuit can be formulated as below:

$$\begin{aligned} P_d &= 1 - \prod_{i=1}^{n}(1 - \alpha_i) \\ &= \sum_i \alpha_i - \sum_{i \neq j} \alpha_i \alpha_j + ... + (-1)^{n-1} \alpha_1 \alpha_2 ... \alpha_n \end{aligned} \tag{3}$$

It indicates that $P_l < P_d < P_u$. Supposing $\alpha_i \ll 1$, we have $P_d \approx P_u$. Accordingly, we have the probability of the error occurrence in a speculation circuit as followings:

$$\begin{aligned} P_1 &= 1 - \sum_{i=0}^{m-1} \alpha_i \\ P_2 &= \sum_{i=0}^{m-1} \alpha_i \\ P_i &= 0, \quad i = 3, ..., m \end{aligned} \tag{4}$$

where P_i denotes the probability to finish the evaluation in i cycles.

According to Equation 2 and 4, the average latency can be expressed as:

$$\tau_{ave} = \sum_{\mu} P_\mu \cdot \mu \cdot \tau_{cyc} = (1 + \sum_{i=0}^{m-1} \alpha_i) \cdot \tau_{cyc} \tag{5}$$

Noting that α_i is a monotonic decreasing function of sampling time t_i^{sa}, the minimal average cycle is obtained when each t_i^{sa}

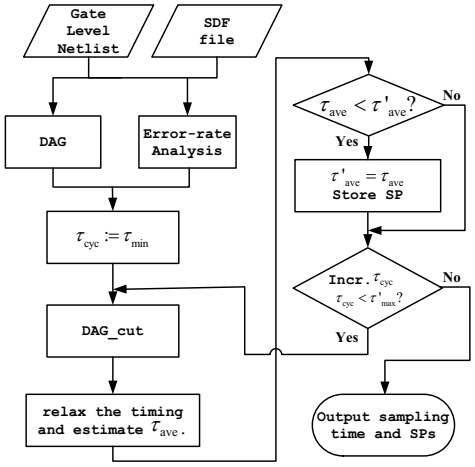

Fig. 3. Design flow

is set to the maximal value satisfying the requirement of τ_{cyc} in Equation 1.

III. Design Flow

This section first formulates the SPs insertion in a combinational circuit as a nonlinear optimization problem, and then we proposed a heuristic algorithm and a design flow to complete the SPs insertion.

A. SPs Insertion Optimization

The performance-oriented SPs insertion in a combinational circuit requires a configuration to minimize the average cycle τ_{ave}:

$$\min \tau_{ave} = C_{ave} * \tau_{cyc} \tag{6}$$

subject to:

$$\begin{aligned} C_{ave} &= f(t_i^{sa}) \qquad i \in SP \\ \tau_{cyc} &= \max_{i \in SP}(t_i^{sa} + \tau_{i,o} + \tau_j^{spec}) \end{aligned} \tag{7}$$

where SP is the set of all the speculation points. Obviously, this is a nonlinear optimization problem. As the value of $f(t_i^{sa})$ can only be obtained by simulations, it is hard to find an analytical solution. Furthermore, in some cases, the area as well as the performance are concerned. Therefore, it is transformed into a multi-objective optimization problem, which is even harder to be solved.

Given a fixed clock cycle, finding the speculation points' set $SP \subset V$, which is on the pareto frontier of the system error rate (which determined the throughput) and area cost, is also too computation costly. In order to reduce the time complexity of selecting the optimal speculation points, we developed a greedy algorithm to partition the stage of the circuit, which can find a possible SP.

B. Proposed Flow

Next, we will proposed a design flow to solve the optimization problem. We break the optimization problem into two steps: the first step will enumerate all possible τ_{cyc} in a user specified region $[\tau_{min}, \tau_{max}]$; the second step tries to find a SPs' insertion to satisfy the τ_{cyc} requirement. The proposed design flow is described in Fig. 3: As we can see, it includes two loops. The outer loop scans τ_{cyc} from τ_{min} to τ_{max} in a specified step. In the inner loop, a greedy algorithm is implemented to insert SPs in a combinational circuit under the τ_{cyc} constraint. This greedy algorithm has a time complexity of $O(|V| + |E|)$, where $|V|, |E|$ is the number of nodes and connections in the circuit.

Algorithm 1 DAG_cut(E, τ_{cyc}, $TimeSpec$)

```
1: TopOrd = Topological_Sort(E)
2: for node = TopOrd(1) to TopOrd(N) do
3:    for each edge (i,j)∈ E do
4:       if AT(i) + τ_{i,j} > AT(j) then
5:          AT(j) = AT(i) + τ_{i,j}
6:       end if
7:       if AT(j) > τ_cyc then
8:          SP = {SP, i}, adding node i to the set of SPs.
9:          AT(i) = TimeSpec(i)
10:         Re-calculate AT(j)
11:      end if
12:   end for
13: end for
```

The algorithm consists of three operations: the initialization operation, the cut operation and the relax operation. In the initialization stage, we obtain the DAG of the original combinational circuit and get the data pair $(\alpha_i(t^{sa0}), t^{sa0})$ of the error rate α and its corresponding sampling time t^{sa0}. All the data pairs are stored in an array $TimeSpec$, which is used in the relax operation of the sampling time. In the cut operation, all the nodes in the DAG are topological sorted, and we calculate the arriving time (AT) of each node. When the AT of a node is greater than the required τ_{cyc}, we replace the parent node causing the violation with a SP. Next, we set the AT of the speculation point as the initial sampling and update all the children nodes. When all the nodes are processed, the algorithm is finished. The detailed algorithm is illustrated in Algorithm 1.

IV. Experiment Result

A. Experiment Setup

A ripple carry adder and several MCNC and ISCAS benchmarks are used to validate the multistage time-domain speculation methodology. We used Design Compiler to synthesize all the circuits and made static timing analysis by PrimeTime under HJTC $0.18\mu m$ standard CMOS technology. 5000 random inputs are used to stimulate the circuis by ModelSim with back-annotation SDF Timing file. We monitored the switching activities of the inner nodes and reported the speculation errors in the VCD file. The average performance of each circuit is obtained by analyzing the simulation results. The highest clock frequency and area overheads are estimated by Design Compiler.

B. Ripple Carry Adder

We deployed the design flow in Section III on a 64-bit ripple carry adder (RCA). The critical path of a RCA is the carry chain. In our experiments, the carry chain is cut into 2, 4 and 8 segments. Accordingly, 1, 3 and 7 SPs are inserted. We scanned the adders with different sampling time under each speculation configuration in Fig. 4. The sampling time and the

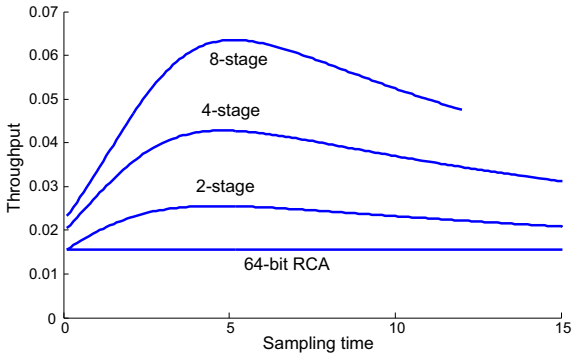

Fig. 4. Multi-stage vs. original 64-bit RCA

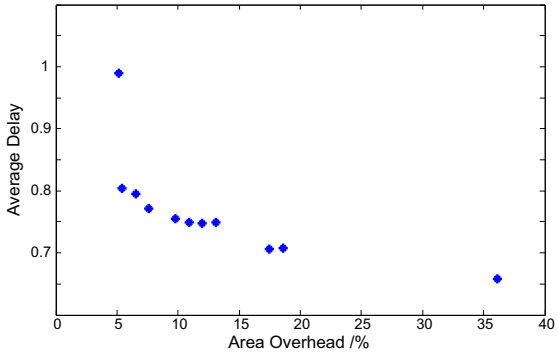

Fig. 5. Performance and area cost's tradeoff, *dalu*

throughput are normalized by τ_{FA} and $1/\tau_{FA}$, where τ_{FA} is the propagation time of a full adder. Fig. 4 shows that there is an optimal sampling time for each adder. It can be explained by the fact that the sampling time affects the throughput by two sides. On one hand, a small t^{sa} leads to a faster clock while satisfying the accuracy of the speculation. On the other hand, a long t^{sa} leads to a more accurate speculation design with a limited peak clock frequency. Given random input patterns, the throughput of the 2-, 4- and 8-stage time-domain speculated adder are 1.79, 2.75, and 4.42 times of the original one, and the area overheads are 2.3%, 7.0% and 16.4%.

C. Some ISCAS and MCNC benchmarks

We also deployed the proposed method to several ISCAS and MCNC benchmarks. The results are obtained under a proper tradeoff between the performance and area overhead and the details are summarized in Table I. It is shown that we obtain 23% average delay reduction with 11% area costs. We provided more detailed results to show the performance-area tradeoff in the design *dalu*. Fig.5 illustrated the Pareto frontier between the performance and the area cost for the design *dalu*.

V. Discussions and Conclusions

In this paper, we proposed a multistage time-domain speculation design methodology, which creates a new paradigm to enhance the average-case performance with reasonable area overheads. The method is proven on a RCA and several ISCAS'85 and MCNC benchmarks. Our future work is to develop a combined time-domain and the functional speculation method, as well as a reconfigurable mechanism varying according to the inputs on the fly.

TABLE I
Experiment Results on ISCAS and MCNC Benchmarks

Netlists	Gates	I/O Ports	SPs	Avg. Delay	Area Ovhd.	Avg. [6] Delay
apex5	404	205	20	0.856	17.6 %	0.874
dalu	274	91	9	0.757	9.85%	0.741
c2670	221	373	1	0.770	1.61%	n.p.
c7552	921	314	40	0.695	13.5 %	0.843

References

[1] M. Choudhury and K. Mohanram, "Masking timing errors on speed-paths in logic circuits," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* IEEE, 2009, pp. 87–92.

[2] J. Sartori and R. Kumar, "Overscaling-friendly timing speculation architectures," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*. ACM, 2010, pp. 209–214.

[3] A. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*. IEEE, 2010, pp. 825–831.

[4] Y. Su, D. Wang, S. Chang, and M. Marek-Sadowska, "An efficient mechanism for performance optimization of variable-latency designs," in *DAC*. ACM New York, NY, USA, 2007, pp. 976–981.

[5] L. Benini, E. Macii, M. Poncino, and G. De Micheli, "Telescopic units: A new paradigm for performance optimization of VLSI designs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 220–232, 2002.

[6] D. Baneres, J. Cortadella, and M. Kishinevsky, "Variable-latency design using function speculation," in *DATE*, 2009.

[7] A. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: a new paradigm for arithmetic circuit design," in *DATE*, 2008, pp. 1250–1255.

[8] Y. Liu, Y. Sun, Z. Y., and H. Yang, "Design Methodology of Variable Latency Adders with Multistage Function Speculation," in *Proceedings of the 11th Int'l Symposium on Quality Electronic Design*. IEEE, 2010, pp. 824–830.

[9] S. Lu, "Speeding up processing with approximation circuits," *Computer*, pp. 67–73, 2004.

[10] Y. Chen, H. Li, J. Li, and C. Koh, "Variable-latency adder (VL-adder): new arithmetic circuit design practice to overcome NBTI," in *ISLPED*. ACM New York, NY, USA, 2007, pp. 195–200.

[11] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, and K. Flautner, "Razor: A low-power pipeline based on circuit-level timing speculation," in *MICRO*. IEEE Computer Society Washington, DC, USA, 2003.